JUNIPER NETWORKS | Engineering Simplicity

Contrail® Networking

Contrail Networking Monitoring and Troubleshooting Guide

Published
2024-01-24

RELEASE
21.4

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® Networking Contrail Networking Monitoring and Troubleshooting Guide
21.4
Copyright © 2024 Juniper Networks, Inc. All rights reserved.

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

About This Guide | xii

1 **Monitoring and Troubleshooting Contrail**

Understanding Contrail Analytics | 2

Understanding Contrail Analytics | 2

Contrail Alert Streaming | 3

Underlay Overlay Mapping in Contrail | 8

- Overview: Underlay Overlay Mapping using Contrail Analytics | 9
- Underlay Overlay Analytics Available in Contrail | 9
- Architecture and Data Collection | 9
- New Processes/Services for Underlay Overlay Mapping | 10
- External Interfaces Configuration for Underlay Overlay Mapping | 11
- Physical Topology | 11
- SNMP Configuration | 12
- Link Layer Discovery Protocol (LLDP) Configuration | 12
- IPFIX and sFlow Configuration | 12
- Sending pRouter Information to the SNMP Collector in Contrail | 14
- pRouter UVEs | 15
- Contrail User Interface for Underlay Overlay Analytics | 17
- Enabling Physical Topology on the Web UI | 17
- Viewing Topology to the Virtual Machine Level | 17
- Viewing the Traffic of any Link | 18
- Trace Flows | 18
- Search Flows and Map Flows | 19
- Overlay to Underlay Flow Map Schemas | 20
- Module Operations for Overlay Underlay Mapping | 23
- SNMP Collector Operation | 23
- Topology Module Operation | 25
- IPFIX and sFlow Collector Operation | 26
- Troubleshooting Underlay Overlay Mapping | 27
- Script to add pRouter Objects | 27

Encryption Between Analytics API Servers and Client Servers | 29

Configuring Contrail Analytics | 32

Analytics Scalability | 32

High Availability for Analytics | 34

vRouter Command Line Utilities | 34

- Overview | 35
- vif Command | 36
- clear Command | 41
- flow Command | 41
- vrfstats Command | 43
- rt Command | 44
- dropstats Command | 45
- mpls Command | 49
- mirror Command | 51
- vxlan Command | 53
- nh Command | 55
- dpdkinfo Command | 58
- dpdkconf Command | 67

Tracing the vRouter Packet Path | 68

- Unicast Packet Path - Intra-VN | 69
- Unicast Packet Path - Inter-VN | 75
- Broadcast, Unknown Unicast, and Multicast Packet Path | 80

Using Contrail Tools | 92

Using Sandump Tool | 94

Security Logging Object | 101

- Defining an SLO | 101
- Attaching an SLO to a Virtual Network and Virtual Machine Interface | 102
  - Attaching an SLO to a Virtual Network | 103
  - Attaching an SLO to a Virtual Machine Interface | 103
- Editing an Existing SLO | 104

System Log Receiver in Contrail Analytics | 104

- Overview | 105

Redirecting System Logs to Contrail Collector | **105**

Exporting Logs from Contrail Analytics | **105**

Sending Flow Messages to the Contrail System Log | **105**

User Configuration for Analytics Alarms and Log Statistics | **107**

Configuring Alarms Based on User-Visible Entities Data | **107**

Examples: Detecting Anomalies | **109**

Configuring the User-Defined Log Statistic | **111**

Implementing the User-Defined Log Statistic | **114**

Contrail Networking Alarms | **117**

Alarms History | **122**

Node Memory and CPU Information | **125**

Role- and Resource-Based Access Control for the Contrail Analytics API | **126**

Configuring Analytics as a Standalone Solution | **127**

Agent Modules in Contrail Networking | **130**

Configuring Secure Sandesh and Introspect for Contrail Analytics | **142**

Configuring Traffic Mirroring to Monitor | 144

Configuring Traffic Analyzers and Packet Capture for Mirroring | **144**

Traffic Analyzer Images | **144**

Configuring Traffic Analyzers | **145**

Setting Up Traffic Mirroring Using Configure > Networking > Services | **145**

Configuring Interface Monitoring and Mirroring | **152**

Mirroring Enhancements | **153**

Analyzer Service Virtual Machine | **155**

Using the Wireshark Plugin to Analyze Packets Between vRouter and vRouter Agent on pkt0 Interface | **158**

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | **163**

Using Contrail Web UI to Monitor and Troubleshoot the Network | 165

Monitoring the System | **165**

Monitor > Infrastructure > Dashboard | **169**

- Monitor Dashboard | **170**
- Monitor Individual Details from the Dashboard | **170**
- Using Bubble Charts | **171**
- Color-Coding of Bubble Charts | **172**

Monitor > Infrastructure > Control Nodes | **173**

- Monitor Control Nodes Summary | **173**
- Monitor Individual Control Node Details | **174**
- Monitor Individual Control Node Console | **176**
- Monitor Individual Control Node Peers | **179**
- Monitor Individual Control Node Routes | **181**

Monitor > Infrastructure > Virtual Routers | **184**

- Monitor vRouters Summary | **184**
- Monitor Individual vRouters Tabs | **186**
- Monitor Individual vRouter Details Tab | **186**
- Monitor Individual vRouters Interfaces Tab | **188**
- Monitor Individual vRouters Networks Tab | **190**
- Monitor Individual vRouters ACL Tab | **191**
- Monitor Individual vRouters Flows Tab | **193**
- Monitor Individual vRouters Routes Tab | **194**
- Monitor Individual vRouter Console Tab | **195**

Monitor > Infrastructure > Analytics Nodes | **198**

- Monitor Analytics Nodes | **198**
- Monitor Analytics Individual Node Details Tab | **200**
- Monitor Analytics Individual Node Generators Tab | **201**
- Monitor Analytics Individual Node QE Queries Tab | **202**
- Monitor Analytics Individual Node Console Tab | **203**

Monitor > Infrastructure > Config Nodes | **206**

- Monitor Config Nodes | **206**
- Monitor Individual Config Node Details | **207**
- Monitor Individual Config Node Console | **208**

Monitor > Networking | **210**

- Monitor > Networking Menu Options | **210**
- Monitor > Networking > Dashboard | **211**

Monitor > Networking > Projects | **213**

Monitor Projects Detail | **214**

Monitor > Networking > Networks | **217**

Query > Flows | **222**

Query > Flows > Flow Series | **223**

Example: Query Flow Series | **226**

Query > Flow Records | **228**

Query > Flows > Query Queue | **231**

Query > Logs | **232**

Query > Logs Menu Options | **233**

Query > Logs > System Logs | **233**

Sample Query for System Logs | **235**

Query > Logs > Object Logs | **237**

Debugging Processes Using the Contrail Introspect Feature | **239**

Example: Debugging Connectivity Using Monitoring for Troubleshooting | **244**

Using Monitoring to Debug Connectivity | **245**

Contrail Analytics Optional Modules | **251**

Using Contrail Command to Monitor and Troubleshoot the Network | 275

Viewing Overlay Routes | **275**

Monitoring Bond Interfaces in DPDK Enabled Devices | **276**

Top N View in Contrail Command | **280**

Contrail Command UI—Top N Feature | **280**

Top N Filter Options | **282**

Chart View | **285**

Viewing Topology Maps from Contrail Command | **286**

Viewing Packet Path in Topology View | **291**

Assign Custom Names to Privileged Ports and VXLAN IDs | **296**

Viewing the Monitoring Dashboards | **302**

Creating a Query for Flows | **306**

Contrail Analytics Optional Modules | 315

Contrail Insights in Contrail Command | 338

Contrail Insights Overview | 338

Contrail Insights Flows in Contrail Command | 339

- Configuring Contrail Insights Flows from Contrail Command | 339
- Configuring Contrail Insights Flows During Fabric Onboarding | 340
- Configuring Contrail Insights Flows by Assigning Telemetry and sFlow Profiles to Devices | 341
- Removing a Telemetry Profile | 351

Viewing Telemetry KPI Alarms for Fabric Devices and Ports | 353

Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command | 364

Adding or Deleting sFlow Collector Nodes by Modifying instances.yml | 378

Configuring Contrail Insights Alarms using Contrail Command | 381

Configuring Instances in Contrail Insights | 406

Viewing Cluster Node Details and Metric Values | 412

Common Support Answers | 417

Debugging Ping Failures for Policy-Connected Networks | 417

Debugging BGP Peering and Route Exchange in Contrail | 425

- Example Cluster | 425
- Verifying the BGP Routers | 425
- Verifying the Route Exchange | 428
- Debugging Route Exchange with Policies | 431
- Debugging Peering with an MX Series Router | 432
- Debugging a BGP Peer Down Error with Incorrect Family | 434
- Configuring MX Peering (iBGP) | 437
- Checking Route Exchange with an MX Series Peer | 439
- Checking the Route in the MX Series Router | 441

Troubleshooting the Floating IP Address Pool in Contrail | 443

- Example Cluster | 444
- Example | 445
- Example: MX80 Configuration for the Gateway | 446
- Ping the Floating IP from the Public Network | 449

Troubleshooting Details | 450
Get the UUID of the Virtual Network | 450
View the Floating IP Object in the API Server | 451
View floating-ips in floating-ip-pools in the API Server | 455
Check Floating IP Objects in the Virtual Machine Interface | 458
View the BGP Peer Status on the Control Node | 462
Querying Routes in the Public Virtual Network | 463
Verification from the MX80 Gateway | 465
Viewing the Compute Node Vnsw Agent | 467
Advanced Troubleshooting | 469

Removing Stale Virtual Machines and Virtual Machine Interfaces | 472

Problem Example | 472
Show Virtual Machines | 474
Delete Methods | 475

Troubleshooting Link-Local Services in Contrail | 476

Overview of Link-Local Services | 476
Troubleshooting Procedure for Link-Local Services | 476
Metadata Service | 478
Troubleshooting Procedure for Link-Local Metadata Service | 478

2 **Contrail Commands and APIs**

Contrail Commands | 481

Getting Contrail Node Status | 481

Overview | 481
UVE for NodeStatus | 482
Node Status Features | 482
Using Introspect to Get Process Status | 489
contrail-status script | 491

contrail-logs (Accessing Log File Messages) | 493

contrail-status (Viewing Node Status) | 496

contrail-version (Viewing Version Information) | 498

Contrail Application Programming Interfaces (APIs) | 501

Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs) | 501

- User-Visible Entities | **502**
- Common UVEs in Contrail | **503**
- Virtual Network UVE | **503**
- Virtual Machine UVE | **504**
- vRouter UVE | **504**
- UVEs for Contrail Nodes | **505**
- Wild Card Query of UVEs | **505**
- Filtering UVE Information | **505**

Log and Flow Information APIs | **515**

- HTTP GET APIs | **515**
- HTTP POST API | **516**
- POST Data Format Example | **516**
- Query Types | **518**
- Examining Asynchronous Query Status | **518**
- Examining Query Chunks | **519**
- Example Queries for Log and Flow Data | **519**

Working with Neutron | **523**

- Data Structure | **523**
- Network Sharing in Neutron | **524**
- Commands for Neutron Network Sharing | **525**
- Support for Neutron APIs | **525**
- Contrail Neutron Plugin | **526**
- DHCP Options | **526**
- Incompatibilities | **527**

Support for Amazon VPC APIs on Contrail OpenStack | **527**

- Overview of Amazon Virtual Private Cloud | **528**
- Mapping Amazon VPC Features to OpenStack Contrail Features | **528**
- VPC and Subnets Example | **529**
- Euca2ools CLI for VPC and Subnets | **530**
- Security in VPC: Network ACLs Example | **530**
- Euca2ools CLI for Network ACLs | **532**
- Security in VPC: Security Groups Example | **532**
- Euca2ools CLI for Security Groups | **533**
- Elastic IPs in VPC | **534**

Euca2ools CLI for Elastic IPs | **534**

Euca2ools CLI for Route Tables | **535**

Supported Next Hops | **535**

Internet Gateway Next Hop Euca2ools CLI | **536**

NAT Instance Next Hop Euca2ools CLI | **536**

Example: Creating a NAT Instance with Euca2ools CLI | **536**

About This Guide

Use this guide to understand Contrail Insights (formerly AppFormix) and Contrail Networking analytics. Contrail Insights provides monitoring availability of the Contrail Networking control plane services. Contrail Networking 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.

Contrail Networking product documentation is organized into multiple guides as shown in Table 1 on page xii , according to the task you want to perform or the deployment scenario.

Table 1: Contrail Networking Guides

Guide Name	Description
Contrail Networking Installation and Upgrade Guide	Provides step-by-step instructions to install and bring up Contrail and its various components.
Contrail Networking for Container Networking Environments User Guide	Provides information about installing and using Contrail Networking in containerized environments using Kubernetes orchestration.
Contrail Networking Fabric Lifecycle Management Guide	Provides information about Contrail underlay management and data center automation.
Contrail Networking and Security User Guide	Provides information about creating and orchestrating highly secure virtual networks.
Contrail Networking Service Provider Focused Features Guide	Provides information about the features that are used by service providers.
Contrail Networking Monitoring and Troubleshooting Guide	Provides information about Contrail Insights and Contrail analytics.

RELATED DOCUMENTATION

- README Access to Contrail Networking Registry 21xx
- Contrail Networking Release Notes 21xx
- Tungsten Fabric Architecture Guide
- Juniper Networks TechWiki: Contrail Networking

PART 1

Monitoring and Troubleshooting Contrail

Understanding Contrail Analytics | 2

Configuring Contrail Analytics | 32

Configuring Traffic Mirroring to Monitor | 144

Using Contrail Web UI to Monitor and Troubleshoot the Network | 165

Using Contrail Command to Monitor and Troubleshoot the Network | 275

Contrail Insights in Contrail Command | 338

Common Support Answers | 417

CHAPTER 1

Understanding Contrail Analytics

IN THIS CHAPTER

- Understanding Contrail Analytics | 2
- Contrail Alert Streaming | 3
- Underlay Overlay Mapping in Contrail | 8
- Encryption Between Analytics API Servers and Client Servers | 29

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 Alert Streaming | 3
- Analytics Scalability | 32
- High Availability for Analytics | 34
- Underlay Overlay Mapping in Contrail | 8
- Monitoring the System | 165
- Debugging Processes Using the Contrail Introspect Feature | 239
- Fat Flows
- System Log Receiver in Contrail Analytics | 104
- Example: Debugging Connectivity Using Monitoring for Troubleshooting | 244

Contrail Alert Streaming

IN THIS SECTION

- Alert API Format | 4
- Analytics APIs for Alerts | 5
- Analytics APIs for SSE Streaming | 6
- Built-in Node Alerts | 6

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/alarms

```
{
    analytics-node: [
        {
            name: "nodec40",
            value: {
                UVEAlarms: {
                    alarms: [
                        {
                            any_of: [
                                {
                                    all_of: [
                                        {
                                            json_operand1_value: ""PROCESS_STATE_STOPPED"",
                                            rule: {
                                                oper: "!=",
                                                operand1: {
                                                    keys: [
                                                        "NodeStatus",
                                                        "process_info",
                                                        "process_state"
                                                    ]
                                                },
                                                operand2: {
                                                    json_value: ""PROCESS_STATE_RUNNING""
                                                }
                                            },
                                            json_vars: {
                                                NodeStatus.process_info.process_name: "contrail-
topology"
                                            }
```

```
                                    }
                                ]
                            }
                        ],
                        severity: 3,
                        ack: false,
                        timestamp: 1457395052889182,
                        token: "eyJ0aW1lc3RhbXAiOiAxNDU3Mzk1MDUyODg5MTgyLCAiaHR0cF9wb3J0I
.............................. jogNTk5NSwgImhvc3RfaXAiOiAiMTAuMjA0LjIxNy4yNCJ9",
                        type: "ProcessStatus"
                    }
                ]
            }
        }
    ]
}
```

In the example:

- An `any_of` attribute contains alarm rules defined in the format `[ [rule1 AND rule2 AND ... AND ruleN] ... OR [rule11 AND rule22 AND ... AND ruleNN] ]`
- 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 | 9
- Underlay Overlay Analytics Available in Contrail | 9
- Architecture and Data Collection | 9
- New Processes/Services for Underlay Overlay Mapping | 10
- External Interfaces Configuration for Underlay Overlay Mapping | 11
- Physical Topology | 11
- SNMP Configuration | 12
- Link Layer Discovery Protocol (LLDP) Configuration | 12
- IPFIX and sFlow Configuration | 12
- Sending pRouter Information to the SNMP Collector in Contrail | 14
- pRouter UVEs | 15
- Contrail User Interface for Underlay Overlay Analytics | 17
- Enabling Physical Topology on the Web UI | 17
- Viewing Topology to the Virtual Machine Level | 17
- Viewing the Traffic of any Link | 18
- Trace Flows | 18
- Search Flows and Map Flows | 19
- Overlay to Underlay Flow Map Schemas | 20
- Module Operations for Overlay Underlay Mapping | 23
- SNMP Collector Operation | 23
- Topology Module Operation | 25
- IPFIX and sFlow Collector Operation | 26
- Troubleshooting Underlay Overlay Mapping | 27
- Script to add pRouter Objects | 27

Overview: Underlay Overlay Mapping using Contrail Analytics

NOTE: This topic applies to Contrail Networking Release 2005 and earlier. Starting in Contrail Networking Release 2008, you can view the path a packet takes in a network. See "Viewing Packet Path in Topology View" on page 291 .

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

Contrail allows you to 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.

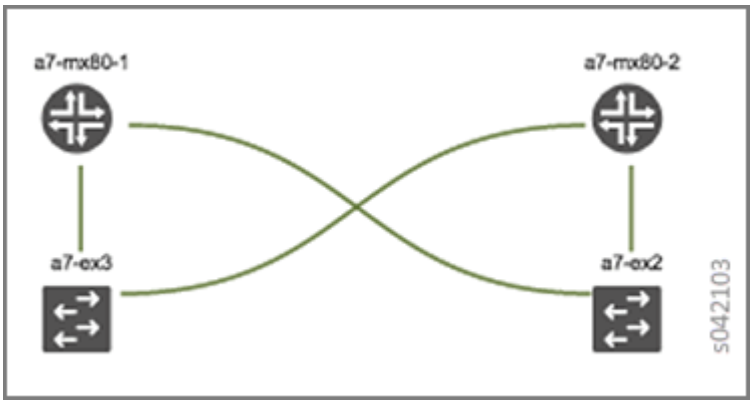
Physical Topology

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 1: 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>
```

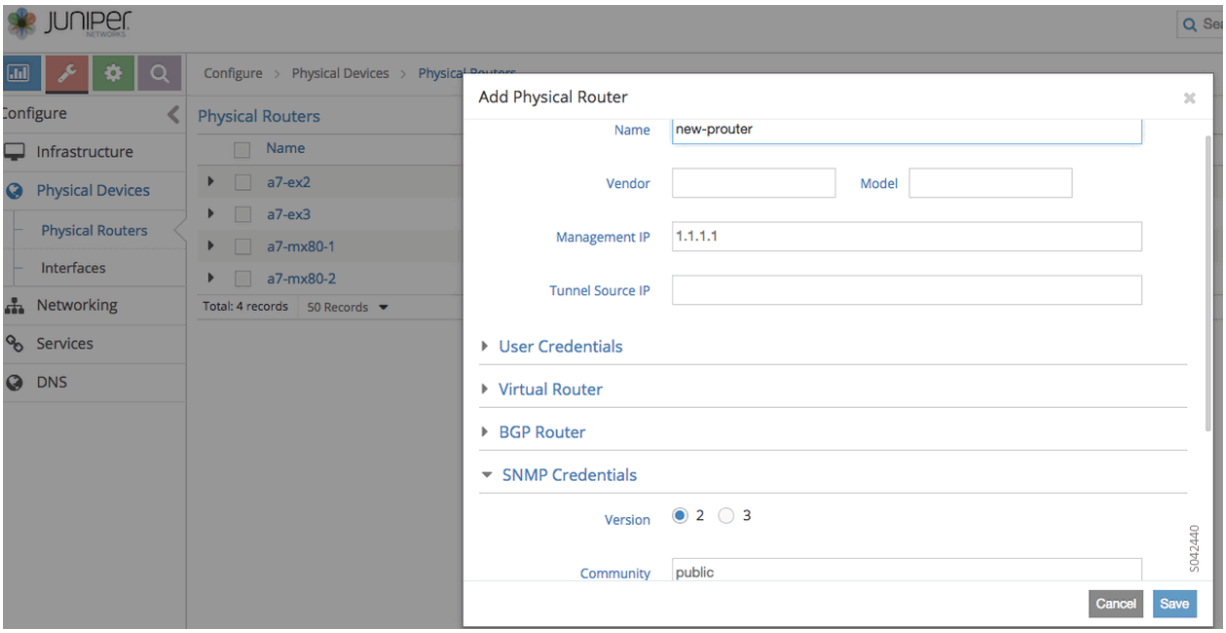
Sending pRouter Information to the SNMP Collector in Contrail

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`.

SNMP collector getting pRouter information from contrail-config

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 2: Add Physical Router Window



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 3: 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"
    }
]
```


S042104

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 4: Sample Output From a pRouter UVE

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;
```

2. Restart webui supervisor.

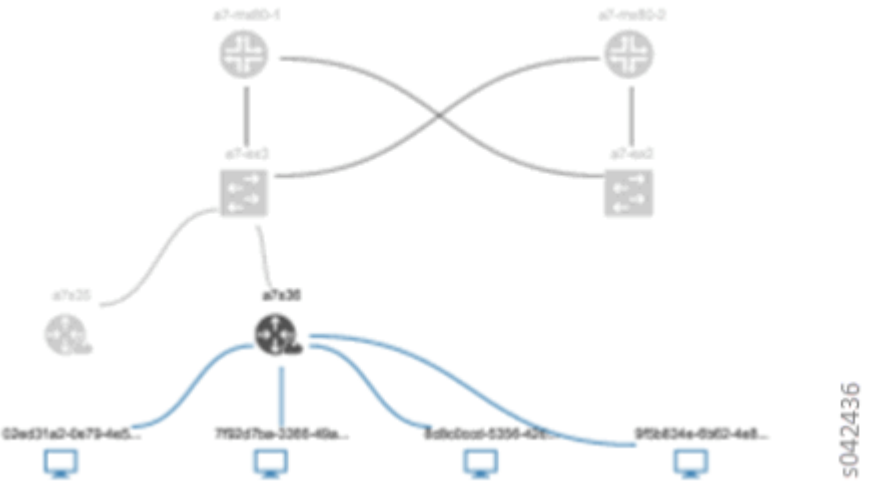
```
service supervisor-webui restart
```

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.

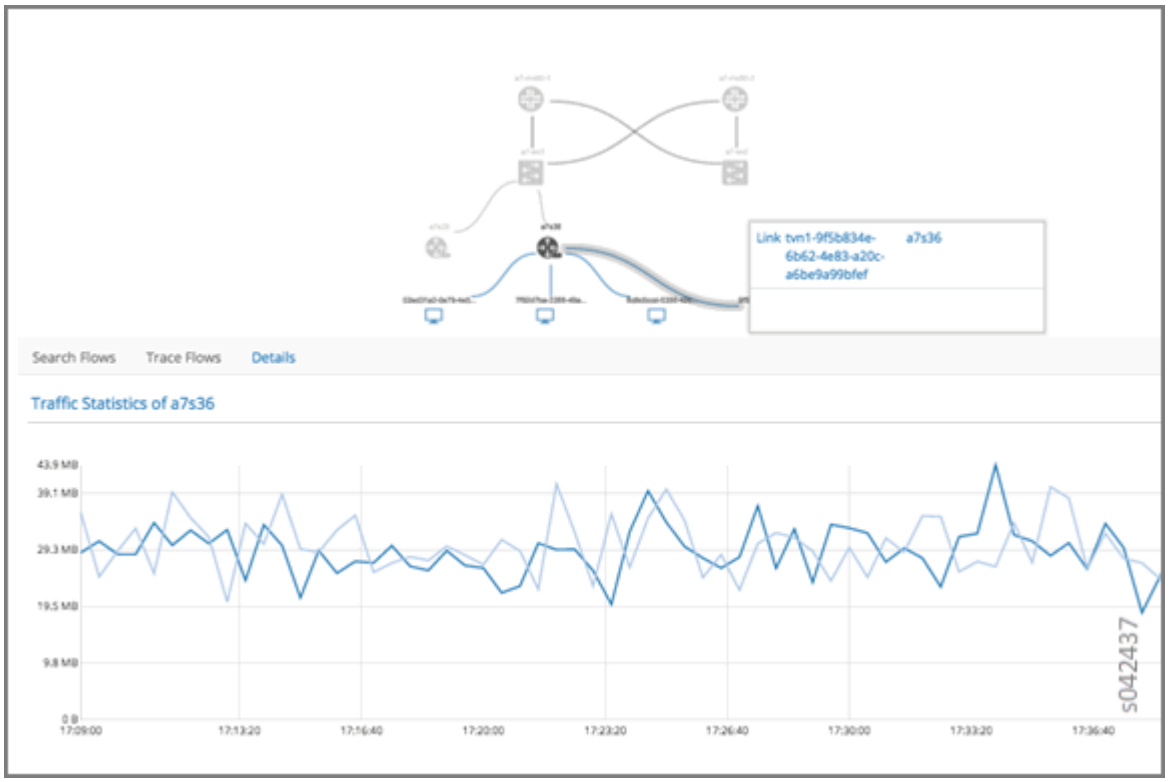
Figure 5: Physical Topology Related to a vRouter



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.

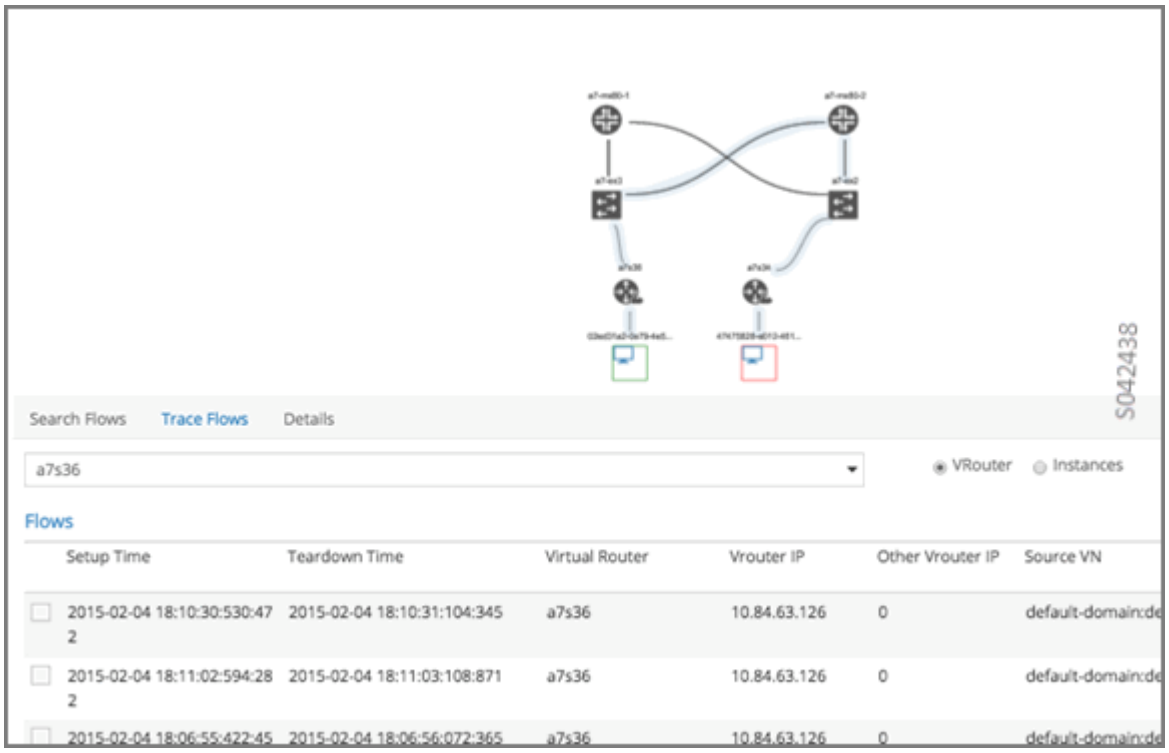
Figure 6: Traffic Statistics Graph



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 7: 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 8: 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 | 2

Contrail Alert Streaming | 3

Encryption Between Analytics API Servers and Client Servers

Contrail Networking Release 1910 supports SSL encryption for the connection between Analytics API servers and Client servers. The Client servers are Service Monitor and Contrail Command, which connects to the Analytics API server through the REST API Port. In releases prior to release 1910, the connection between Analytics API servers and the Client servers was not encrypted, which could pose a security threat.

SSL encryption is supported in Contrail Networking Release 1910 only when Contrail Networking is deployed with Red Hat OpenStack Platform (RHOSP). In the RHOSP deployment, a global flag is added, which determines the status of the SSL encryption.

If the global flag is enabled:

- You do not have to modify the configuration files as SSL encryption is automatically enabled.
- You must modify the configuration files if you want to disable SSL encryption.

If the global flag is disabled:

- You do not have to modify the configuration files as SSL encryption is automatically disabled.
- You cannot enable SSL encryption, even if you modify the configuration files. The certificates are not generated during deployment as the global flag is disabled.

The configuration files are `contrail-analytics-api.conf`, `contrail-svc-monitor.conf`, and `command_servers.yml`. In the configuration files, modify the following parameters in the Table 2 on page 30 below to enable or disable SSL based encryption:

Table 2: SSL Encryption Parameters

Parameters	Description	Default
analytics_api_ssl_enable	Enables or disables support for SSL encryption between Analytics API server and Client server.	If the value is assigned **TRUE**: Support for SSL encryption is enabled. If the value is assigned **FALSE**: Support for SSL encryption is not enabled and the Analytics API server does not accept HTTPS requests.
analytics_api_insecure_enable	Enables or disables support for required certificates in HTTPS requests.	If the value is assigned **TRUE**: HTTPS connection is supported without the certificates. If the value is assigned **FALSE**: HTTPS connection is not supported without the certificates.
analytics_api_ssl_keyfile	Path to the node's private key.	**/etc/contrail/ssl/private/server-privkey.pem**
analytics_api_ssl_certfile	Path to the node's public certificate.	**/etc/contrail/ssl/certs/server.pem**
analytics_api_ssl_ca_cert	Path to the CA certificate	**/etc/ipa/ca.crt**

Once these parameters are configured, the Analytics API server starts using SSL certificates, which enables SSL encryption support for connection between Analytics API servers and Client servers.

Change History Table

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

Release	Description
1910	Contrail Networking Release 1910 supports SSL encryption for the connection between Analytics API servers and Client servers.

CHAPTER 2

Configuring Contrail Analytics

IN THIS CHAPTER

- Analytics Scalability | 32
- High Availability for Analytics | 34
- vRouter Command Line Utilities | 34
- Tracing the vRouter Packet Path | 68
- Using Contrail Tools | 92
- Using Sandump Tool | 94
- Security Logging Object | 101
- System Log Receiver in Contrail Analytics | 104
- Sending Flow Messages to the Contrail System Log | 105
- User Configuration for Analytics Alarms and Log Statistics | 107
- Contrail Networking Alarms | 117
- Alarms History | 122
- Node Memory and CPU Information | 125
- Role- and Resource-Based Access Control for the Contrail Analytics API | 126
- Configuring Analytics as a Standalone Solution | 127
- Agent Modules in Contrail Networking | 130
- Configuring Secure Sandesh and Introspect for Contrail Analytics | 142

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).

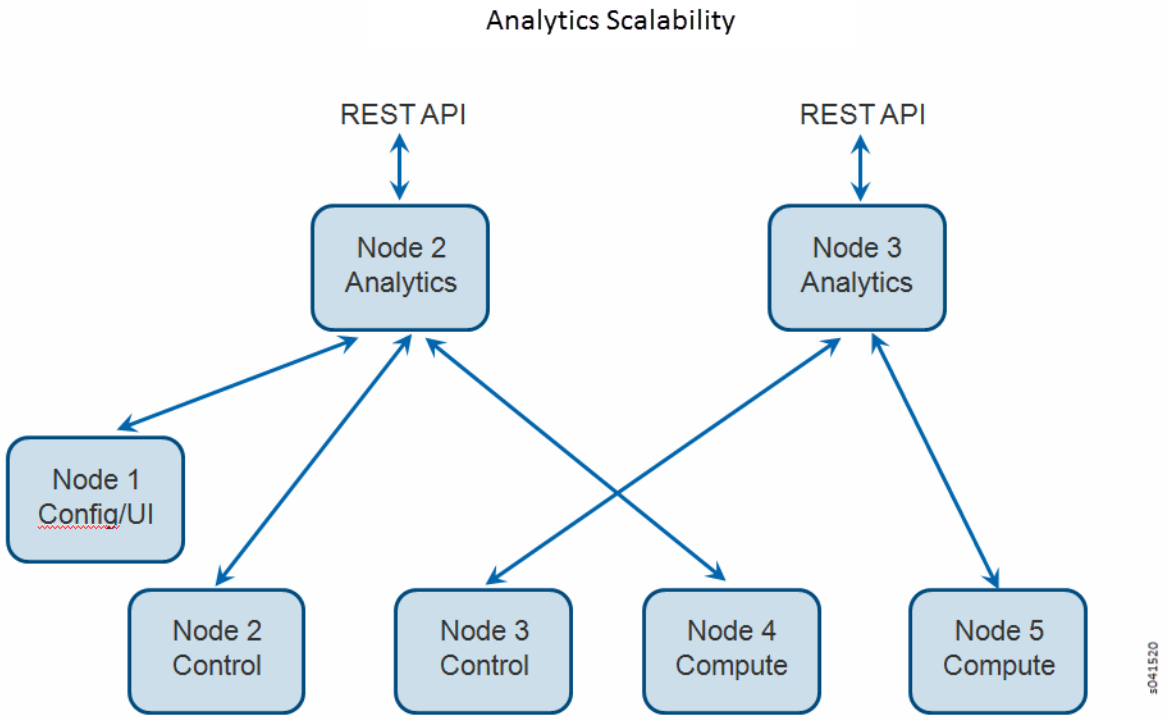
This section provides a brief description of the recommended configuration of analytics in Contrail to achieve horizontal scalability.

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 9 on page 33 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 9: 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.

vRouter Command Line Utilities

IN THIS SECTION

Overview | 35

- vif Command | 36
- clear Command | 41
- flow Command | 41
- vrfstats Command | 43
- rt Command | 44
- dropstats Command | 45
- mpls Command | 49
- mirror Command | 51
- vxlan Command | 53
- nh Command | 55
- dpdkinfo Command | 58
- dpdkconf Command | 67

Overview

vRouter is the component that takes packets from VMs and forwards them to their destinations. In this effort, vRouter depends on the vRouter agent to make sense of the overall topology, understand the various policies that govern the communication between VMs, and program them in vRouter in a way vRouter understands.

vRouter has a few fundamental data structures that abstracts out the various communication paths. There is "interface," "flow," "route," and "nexthop" that enables vRouter to push packets to their eventual destinations. In addition, vRouter also has good statistics that can help understand and debug packet paths. Various command line utilities provided by the vRouter can be used to display these data structures and better understand the behavior that one sees in a compute node.

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.

(Continued)

Command	Description
`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.
`dpdkinfo`	Displays internal data structure details of a DPDK enabled vRouter.
`dpdkconf`	Use this command to add or delete a DDP profile.

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.

The `vif` command can be used to see the interfaces that the vRouter is aware of by including the `--list` option.

Example: vif --list

```
bash$ vif --list
Vrouter Interface Table

Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
       Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
       D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
       Mnp=No MAC Proxy

vif0/0      OS: eth0 (Speed 1000, Duplex 1)
            Type:Physical HWaddr:00:25:90:c3:08:68 IPaddr:0
            Vrf:0 Flags:L3L2Vp MTU:1514 Ref:22
            RX packets:2664341  bytes:702708970 errors:0
            TX packets:1141456  bytes:234609942 errors:0

vif0/1      OS: vhost0
            Type:Host HWaddr:00:25:90:c3:08:68 IPaddr:0
            Vrf:0 Flags:L3L2 MTU:1514 Ref:3
            RX packets:716612  bytes:155442906 errors:0
            TX packets:2248399  bytes:552491888 errors:0

vif0/2      OS: pkt0
            Type:Agent HWaddr:00:00:5e:00:01:00 IPaddr:0
            Vrf:65535 Flags:L3 MTU:1514 Ref:2
            RX packets:450524  bytes:94618532 errors:0
            TX packets:437968  bytes:66753290 errors:0

vif0/3      OS: tap519615d8-a2
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0
            Vrf:1 Flags:PL3L2 MTU:9160 Ref:6
            RX packets:134  bytes:15697 errors:0
            TX packets:8568  bytes:945944 errors:0
```

Table 3: 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`	`Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0` 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: • 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 3: vif Fields *(Continued)*

vif Output Field	Description
`Vrf:xxxxx`	`Vrf:65535 Flags:L3 MTU:1514 Ref:2` 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). Flag options identify that the following are enabled for the interface: • P - Policy. All traffic that comes to vRouter from this interface are subjected to 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. All packets that egresses this interface are mirrored. • Mr - Mirroring receive direction. All packets that ingresses this interface will be mirrored. • Tc - Checksum offload on the transmit side. Valid only on the physical interface.
`Rx`	`RX packets:60 bytes:4873 errors:0` Packets received by vRouter from this interface.
`Tx`	`TX packets:21 bytes:2158 errors:0` Packets transmitted out by vRouter on this interface.

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  <C>]
           [--add <C>> --mac <mac> --vrf <vrf>
                 --type [vhost|agent|physical|virtual|monitoring]
                 --transport [eth|pmd|virtual|socket]
                 --xconnect <physical interface name>
                 --policy, --vhost-phys, --dhcp-enable]
                 --vif <vif ID> --id <intf_id> --pmd --pci]
           [--delete <intf_id>|<intf_name>]
           [--get <intf_id>][--kernel]
           [--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
           [--list][--core <core number>][--rate]
           [--sock-dir <sock dir>]
           [--clear][--id <intf_id>][--core <core_number>]
           [--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> i` is the vRouter interface ID as given by `vif0/X`, where `X` is the ID. So, in `vif0/1`, `1` is the interface index of that vif inside the vRouter module.
`--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 is the kernel ID.

(Continued)

Option	Description
`--set`	Set working parameters of an interface. The 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 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.
`--clear`	Clears statistics for all interfaces on all cores. For more information, see "clear Command" on page 41 .

clear Command

Contrail Networking Release 2008 supports clearing of vif statistics counters for all interfaces by using the `--clear` command. For more information on `--clear` command options, see Table 4 on page 41 .

Table 4: clear Command Options

Option	Description
`--clear`	Clears statistics for all interfaces on all cores.
`--clear --id <vif-id>`	Clears statistics for a specific interface.
`--clear --core <core-id>`	Clears statistics on a specific core for all interfaces.
`--clear --id <vif-id> --core <core-id>`	Clears statistics for a specific interface on a specific core.

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 -1 is the only relevant debugging option.

```
  # flow -l
Flow table
   Index        Source:Port                  Destination:Port   Proto(V)

-----------------------------------------------------------------------------------------

 263484          1.1.1.252:1203            1.1.1.253:0        1 (3)
                    (Action:F, S(nh):91,  Statistics:22/1848)
     379480          1.1.1.253:1203            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 (V) 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 vRouter maintains 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 ‘mtrie’.

The `Flags` field can have two values. `L` indicates that the label field is valid, and `H` indicates that `vroute` 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. Use the `dropstats --debug` command to view the `Cloned Original` counters.

Example: dropstats

```

(vrouter-agent-dpdk)[root@nodec56 /]$ dropstats
```

```
Invalid IF                    0
Trap No IF                    0
IF TX Discard                 0
IF Drop                       0
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
New Flow Drops                0
Flow Unusable (Eviction)      0

Original Packet Trapped       0

Discards                      0
TTL Exceeded                  0
Mcast Clone Fail              0

Invalid NH                    2
Invalid Label                 0
Invalid Protocol              0
Etree Leaf to Leaf            0
Bmac/ISID Mismatch            0
Rewrite Fail                  0
Invalid Mcast Source          0
Packet Loop                   0

Push Fails                    0
Pull Fails                    0
Duplicated                    0
Head Alloc Fails              0
PCOW fails                    0
Invalid Packets               0

Misc                          0
Nowhere to go                 0
Checksum errors               0
No Fmd                        0
```

```
Invalid VNID                  0
Fragment errors               0
Invalid Source                0
Jumbo Mcast Pkt with DF Bit   0
No L2 Route                   0
Memory Failures               0
Fragment Queueing Failures    0
No Encrypt Path Failures      0
Invalid HBS received packet   0

VLAN fwd intf failed TX       0
VLAN fwd intf failed enq      0

(vrouter-agent-dpdk)[root@nodec56 /]$ dropstats --debug
Cloned Original               0
```

NOTE: `Cloned Original` drops are still included in the `Drops` section in the output of the `vif --list` command.

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.
`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.

(Continued)

Next Hop Type	Description
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
```

dpdkinfo Command

In Contrail Networking Release 2008, the `dpdkinfo` command enables you to see the details of the internal data structures of a DPDK enabled vRouter.

dpdkinfo Options

Use `dpdkinfo --help` to display all options available for the `dpdkinfo` command. The `dpdkinfo` command options are described in the following table:

Option	Description
`--bond`	Displays the bond interface information for primary and backup devices in a bond interface.
`--lacp all`	Displays the Link Aggregation Control Protocol (LACP) configuration for Slow and Fast LACP timers along with port details of actor and partner interfaces in a LACP exchange.
`--mempool all`	Displays summary of used and available memory buffers from all memory pools.
`--mempool <mempool_name>`	Displays information about the specified memory pool.
`--stats eth`	Displays NIC statistics information for the packets received (Rx) and transmitted (Tx) by the vRouter.
`--xstats all`	Displays extended NIC statistics information from NIC cards.
`--xstats=<interface-id>`	Displays extended NIC information of the primary and backup devices for the given interface-id (Primary->0, Slave_0->1, Slave_1 ->2).

(Continued)

Option	Description
`--lcore`	Displays the Rx queue mapped interfaces along with Queue ID.
`--app`	Displays the overall application information like actual physical interface name, number of cores, VLAN, queues, and so on.
`dpdkinfo --ddp list`	Displays the list of DDP profiles added in the vRouter.

Example: dpdkinfo --bond

The `dpdkinfo --bond` displays the following information for primary and backup devices: actor/partner status, actor/partner key, actor/partner system priority, actor/partner MAC address, actor/partner port priority, actor/partner port number, and so on.

```
dpdkinfo --bond
No. of bond slaves: 2
Bonding Mode: 802.3AD Dynamic Link Aggregation
Transmit Hash Policy: Layer 3+4 (IP Addresses + UDP Ports) transmit load balancing
MII status: UP
MII Link Speed: 1000 Mbps
MII Polling Interval (ms): 10
Up Delay (ms): 0
Down Delay (ms): 0
Driver: net_bonding

802.3ad info :
LACP Rate: slow
Aggregator selection policy (ad_select): Stable
System priority: 32512
System MAC address:00:50:00:00:00:00
Active Aggregator Info:
        Aggregator ID: 0
        Number of ports: 2
        Actor Key: 4096
        Partner Key: 0
        Partner Mac Address: 00:00:80:7a:9b:05

Slave Interface(0): 0000:02:00.0
```

```
Slave Interface Driver: net_ixgbe
MII status: DOWN
MII Link Speed: 0 Mbps
Permanent HW addr:00:aa:7b:93:00:00
Aggregator ID: 13215
Duplex: half
Bond MAC addr:ac:1f:6b:a5:0f:de
Details actor lacp pdu:
        system priority: 0
        system mac address:00:aa:7b:93:00:00
        port key: 0
        port priority: 0
        port number: 63368
        port state: 0 ()

Details partner lacp pdu:
        system priority: 15743
        system mac address:00:00:80:01:9c:05
        port key: 0
        port priority: 0
        port number: 28836
        port state: 117 (ACT AGG COL DIST DEF )

Slave Interface(1): 0000:02:00.1
Slave Interface Driver: net_ixgbe
MII status: UP
MII Link Speed: 1000 Mbps
Permanent HW addr:ac:1f:6b:a5:0f:df
Aggregator ID: 1
Duplex: full
Bond MAC addr:ac:1f:6b:a5:0f:df
Details actor lacp pdu:
        system priority: 65535
        system mac address:ac:1f:6b:a5:0f:df
        port key: 17
        port priority: 255
        port number: 2
        port state: 61 (ACT AGG SYNC COL DIST )

Details partner lacp pdu:
        system priority: 127
        system mac address:ec:3e:f7:5f:f0:40
        port key: 3
```

```
        port priority: 127
        port number: 10
        port state: 63 (ACT TIMEOUT AGG SYNC COL DIST )
```

Example: dpdkinfo --lacp all

The `dpdkinfo --lacp all` command displays the following information for primary devices: LACP rate and LACP configuration details, which include Fast periodic (ms), Slow periodic (ms), Short timeout (ms), Long timeout (ms), LACP packet statistics for Tx and Rx counters, and so on. Also, `dpdkinfo --lacp all` displays actor and partner port status details of all the backup devices.

```
dpdkinfo --lacp all
LACP Rate: fast

Fast periodic (ms): 900
Slow periodic (ms): 29000
Short timeout (ms): 3000
Long timeout (ms): 90000
Aggregate wait timeout (ms): 2000
Tx period (ms): 500
Update timeout (ms): 100
Rx marker period (ms): 2000

Slave Interface(0): 0000:04:00.0
Details actor lacp pdu:
       port state: 63 (ACT TIMEOUT AGG SYNC COL DIST )

Details partner lacp pdu:
       port state: 61 (ACT AGG SYNC COL DIST )

Slave Interface(1): 0000:04:00.1
Details actor lacp pdu:
       port state: 63 (ACT TIMEOUT AGG SYNC COL DIST )

Details partner lacp pdu:
       port state: 61 (ACT AGG SYNC COL DIST )

LACP Packet Statistics:
              Tx     Rx
0000:04:00.0  6      28
0000:04:00.1  7      30
```

Example: dpdkinfo --mempool all and dpdk --mempool <mempool-name>

The `dpdkinfo --mempool all` displays a summary of the memory pool information of the primary and backup devices, which include number of available memory pools, size of the memory pool, and so on.

The `dpdk --mempool <mempool-name>` displays detailed information of the memory pool you have specified in the command.

```
dpdkinfo --mempool all

-------------------------------------------------
Name                 Size       Used     Available
-------------------------------------------------
rss_mempool          16384       620       15765
frag_direct_mempool   4096        0         4096
frag_indirect_mempool 4096        0         4096
slave_port0_pool      8193        0         8193
packet_mbuf_pool      8192        4         8188
slave_port1_pool      8193       125        8068

 dpdkinfo --mempool rss_mempool
rss_mempool
flags = 10
nb_mem_chunks = 77
size = 16384
populated_size = 16384
header_size = 64
elt_size = 9648
trailer_size = 80
total_obj_size = 9792
private_data_size = 64
avg bytes/object = 9856.000000
Internal cache infos:
        cache_size=256
        cache_count[0]=65
        cache_count[8]=219
        cache_count[9]=2
        cache_count[10]=156
        cache_count[11]=195
total_cache_count=637
common_pool_count=15137
```

Example: dpdkinfo --stats eth

The `dpdkinfo --stats eth` command reads Rx and Tx packets statistics from the NIC card and displays the information.

```
dpdkinfo --stats eth
Master Info:
RX Device Packets:1289, Bytes:148651, Errors:0, Nombufs:0
Dropped RX Packets:0
TX Device Packets:2051, Bytes:237989, Errors:0
Queue Rx: [0]1289
      Tx: [0]2051
      Rx Bytes: [0]148651
      Tx Bytes: [0]234429
      Errors:
------------------------------------------------------------------------

Slave Info(0000:02:00.0):
RX Device Packets:0, Bytes:0, Errors:0, Nombufs:0
Dropped RX Packets:0
TX Device Packets:0, Bytes:0, Errors:0
Queue Rx:
      Tx:
      Rx Bytes:
      Tx Bytes:
      Errors:
------------------------------------------------------------------------

Slave Info(0000:02:00.1):
RX Device Packets:1289, Bytes:148651, Errors:0, Nombufs:0
Dropped RX Packets:0
TX Device Packets:2051, Bytes:237989, Errors:0
Queue Rx: [0]1289
      Tx: [0]2051
      Rx Bytes: [0]148651
      Tx Bytes: [0]234429
      Errors:
```

Example: dpdkinfo --xstats

The `dpdkinfo --xstats` command reads the Rx and Tx from the NIC cards and displays the packet statistics in detail.

```
dpdkinfo --xstats
Master Info:
Rx Packets:
        rx_good_packets: 1459
        rx_q0packets: 1459
Tx Packets:
        tx_good_packets: 2316
        tx_q0packets: 2316
Rx Bytes:
        rx_good_bytes: 161175
        rx_q0bytes: 161175
Tx Bytes:
        tx_good_bytes: 265755
        tx_q0bytes: 261915
Errors:
Others:
-------------------------------------------------------------------

Slave Info(0):0000:02:00.0
Rx Packets:
Tx Packets:
Rx Bytes:
Tx Bytes:
Errors:
        mac_local_errors: 2
Others:
-------------------------------------------------------------------

Slave Info(1):0000:02:00.1
Rx Packets:
        rx_good_packets: 1459
        rx_q0packets: 1459
        rx_size_64_packets: 677
        rx_size_65_to_127_packets: 641
        rx_size_128_to_255_packets: 54
        rx_size_256_to_511_packets: 48
        rx_size_512_to_1023_packets: 3
        rx_size_1024_to_max_packets: 36
        rx_broadcast_packets: 3
```

```
        rx_multicast_packets: 772
        rx_total_packets: 1461
Tx Packets:
        tx_good_packets: 2316
        tx_q0packets: 2316
        tx_total_packets: 2316
        tx_size_64_packets: 276
        tx_size_65_to_127_packets: 582
        tx_size_128_to_255_packets: 1433
        tx_size_256_to_511_packets: 4
        tx_size_512_to_1023_packets: 3
        tx_size_1024_to_max_packets: 18
        tx_multicast_packets: 1431
        tx_broadcast_packets: 9
Rx Bytes:
        rx_good_bytes: 161175
        rx_q0bytes: 161175
        rx_total_bytes: 161567
Tx Bytes:
        tx_good_bytes: 265755
        tx_q0bytes: 261915
Errors:
        mac_local_errors: 2
Others:
        out_pkts_untagged: 2316
```

Example: dpdkinfo --lcore

The `dpdkinfo --lcore` displays Logical core (lcore) information, which includes number of forwarding lcores, the interfaces mapped to the lcore, and queue-ID of the interfaces.

```
dpdkinfo --lcore
No. of forwarding lcores: 2
No. of interfaces: 4
Lcore 0:
        Interface: bond0.102           Queue ID: 0
        Interface: vhost0              Queue ID: 0

Lcore 1:
        Interface: bond0.102           Queue ID: 1
        Interface: tapd1b53efb-9e      Queue ID: 0
```

dpdkinfo --app

The `dpdkinfo --app` command displays the following information:

- Application related information about number of lcores, the names of the existing backup interfaces, and so on.
- For VLAN configured devices the command displays VLAN name, tag, and vlan_vif name.
- For bond interfaces the command displays ethdev information, which include Max rx queues, Max tx queues, Reta size, Port id, number of ethdev slaves, Tapdev information, and so on.
- Monitoring interface names (if available) and SR-IOV information, which includes logical core, ethdev port ID, and driver name.

```
dpdkinfo --app
No. of lcores: 12
No. of forwarding lcores: 2
Fabric interface: bond0.102
Slave interface(0): enp2s0f0
Slave interface(1): enp2s0f1
Vlan name: bond0
Vlan tag: 102
Vlan vif: bond0
Ethdev (Master):
        Max rx queues: 128
        Max tx queues: 64
        Ethdev nb rx queues: 2
        Ethdev nb tx queues: 64
        Ethdev nb rss queues: 2
        Ethdev reta size: 128
        Ethdev port id: 2
        Ethdev nb slaves: 2
        Ethdev slaves: 0 1 0 0 0 0

Ethdev (Slave 0): 0000:02:00.0
        Nb rx queues: 2
        Nb tx queues: 64
        Ethdev reta size: 128

Ethdev (Slave 1): 0000:02:00.1
        Nb rx queues: 2
        Nb tx queues: 64
        Ethdev reta size: 128
```

```
Tapdev:
        fd: 39 vif name: bond0
        fd: 48 vif name: vhost0
```

Example: dpdkinfo --ddp list

In Contrail Networking Release 2011, you can use the `dpdkinfo --ddp list` command to display the list of DDP profiles added in the vRouter.

The `dpdkinfo --ddp list` displays a summary of the DDP profile added in the vRouter. The summary of the profile information includes tracking ID of the profile, version number, and profile name.

```
(contrail-tools)[root@cs-scale-02 /]$ dpdkinfo --ddp list
Profile count is: 1

Profile 0:
Track id:     0x8000000c
Version:      1.0.0.0
Profile name: L2/L3 over MPLSoGRE/MPLSoUDP
```

dpdkconf Command

In Contrail Networking Release 2011, the `dpdkconf` command enables you to configure a DPDK enabled vRouter. In release 2011, you can use the `dpdkconf` command to enable or delete a DDP profile in vRouter.

Example: dpdkconf --ddp add

Use the `dpdkconf --ddp add` command during runtime to enable a DDP profile in a DPDK enabled vRouter.

```
(contrail-tools)[root@cs-scale-02 /]$ dpdkconf --ddp add
Programming DDP image mplsogreudp - success
```

Example: dpdkconf --ddp delete

Use the `dpdkconf --ddp delete` command to delete a DDP profile, which is already loaded in the vRouter.

```
(contrail-tools)[root@cs-scale-02 /]$ dpdkconf --ddp delete
vr_dpdk_ddp_del: Removed DDP image mplsogreudp - success
```

Change History Table

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

Release	Description
2011	In Contrail Networking Release 2011, you can use the `dpdkinfo --ddp list` command to display the list of DDP profiles added in the vRouter.
2011	In Contrail Networking Release 2011, the `dpdkconf` command enables you to configure a DPDK enabled vRouter. In release 2011, you can use the `dpdkconf` command to enable or delete a DDP profile in vRouter.
2008	Contrail Networking Release 2008 supports clearing of vif statistics counters for all interfaces by using the `--clear` command.
2008	In Contrail Networking Release 2008, the `dpdkinfo` command enables you to see the details of the internal data structures of a DPDK enabled vRouter.

Tracing the vRouter Packet Path

IN THIS SECTION

- Unicast Packet Path - Intra-VN | **69**
- Unicast Packet Path - Inter-VN | **75**
- Broadcast, Unknown Unicast, and Multicast Packet Path | **80**

Contrail Networking vRouter is the component that takes packets from virtual machines (VM)s and forwards them to their destinations. Tracing is a useful tool for debugging the packet path.

In this topic, we trace the vRouter packet path in the following use cases:

- Unicast Packet Path - Intra-VN
- Unicast Packet Path - Inter-VN
- Broadcast, Unknown Unicast, and Multicast Packet Path

Unicast Packet Path - Intra-VN

IN THIS SECTION

- Intra-Compute Use Case | 69
- Inter-Compute Use Case | 72

This procedure steps through debugging the unicast packet path for intra-virtual network (intra-VN) traffic from VM1 to VM2 (on same compute node) and VM3 (on different compute node). In this example, the VMs listed are in the same subnet `10.1.1.0/24`. Intra-VN traffic is within the same virtual network.

VM1 IP address `10.1.1.5/32` (Compute 1)

VM2 IP address `10.1.1.6/32` (Compute 1)

VM3 IP address `10.1.1.7/32` (Compute 2)

Intra-Compute Use Case

1. Discover the `vif` interfaces corresponding to the virtual machine interfaces (VMI)s of the VM by using the command:

   ```
   vif --list
   ```

   You can also discover the `vif` interfaces by entering the introspect URL.

   Example:

   ```
   http://10.1.1.5:8085/Snh_ItfReq?
   name=&type=&uuid=&vn=&mac=&ipv4_address=&ipv6_address=&parent_uuid=&ip_active=&ip6_active=&l2_
   active=
   ```

   NOTE: Replace the IP address with the actual compute IP address in the introspect HTTP URL.

2. Run the `vif --get <index>` command to verify the virtual routing and forwarding (VRF) and Policy flags are set in the vRouter interface (VIF).

   Example output verifying flags for each `vif`:

```
vif0/4      OS: tapdd789d34-27
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.5
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:30  bytes:8676 errors:0
            TX packets:170  bytes:7140 errors:0
            ISID: 0 Bmac: 02:dd:78:9d:34:27
            Drops:81

vif0/6      OS: tapaedbc037-bf
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.6
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:96316  bytes:4858043 errors:0
            TX packets:96562  bytes:4884177 errors:0
            ISID: 0 Bmac: 02:ae:db:c0:37:bf
            Drops:2
```

3. Run the following command to display all of the entries from the bridge table:

```
rt --dump <vrf id> --family bridge
```

   Example:

```
rt --dump 2 --family bridge
Flags: L=Label Valid, Df=DHCP flood, Mm=Mac Moved, L2c=L2 Evpn Control Word, N=New Entry,
Ec=EvpnControlProcessing
vRouter bridge table 0/2
Index       DestMac                  Flags           Label/VNID      Nexthop           Stats
31264       0:0:5e:0:1:0             Df                   -                3          206834
79784       2:dd:78:9d:34:27         Df                   -               44               4
112924      ff:ff:ff:ff:ff:ff        LDf                  5               48              35
115240      2:0:0:0:0:2              Df                   -               12               0
169408      2:ae:db:c0:37:bf         Df                   -               25               1
183944      2:99:ef:64:96:e1         LDf                 27               20               0
205564      2:0:0:0:0:1              Df                   -               12               0
252380      0:25:90:c5:58:94         Df                   -                3               0
```

Highlighted in the example is the destination MAC address of the destination VM in the bridge table and the next-hop identifier associated with it.

4. Run `nh --get <nh id>` to display the next-hop details.

   Example:

```
nh --get 25
Id:25         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:3          Vrf:2
              Flags:Valid, Policy, Etree Root,
              EncapFmly:0806 Oif:6 Len:14
              Encap Data: 02 ae db c0 37 bf 00 00 5e 00 01 00 08 00
```

   In the example, `Oif:6` is the OIF index in the next hop which is the outgoing interface for the packet. The `Encap Data` corresponds to the L2 encapsulation that is added to the IP packet before the packet is forwarded to the outgoing interface.

5. Run `vif --get <oifindex>` to get the outgoing VIF details.

   Example:

```
vif --get 6
vif0/6      OS: tapaedbc037-bf
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.6
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:96935  bytes:4892936 errors:0
            TX packets:97235  bytes:4921004 errors:0
            ISID: 0 Bmac: 02:ae:db:c0:37:bf
            Drops:2
```

   The received packet `RX` and transmitted packet `TX` counters for the corresponding VIF interfaces are incremented when traffic is flowing through.

6. Run the `flow -l` command to list the flows created. If the Policy flag is enabled on the VIFs, a flow is created as shown in the example.

   Example: Ping `10.1.1.6` from `10.1.1.5`.

```
flow -l
Index               Source:Port/Destination:Port                    Proto(V)
-----------------------------------------------------------------------------------
   876<=>1020         10.1.1.6:1599                                      1 (2)
                      10.1.1.5:0
```

```
(Gen: 12, K(nh):21, Action:F, Flags:, QOS:-1, S(nh):21,  Stats:9/882,
 SPort 54920, TTL 0, Sinfo 6.0.0.0)

  1020<=>876           10.1.1.5:1599                                    1 (2)
                       10.1.1.6:0
(Gen: 2, K(nh):29, Action:F, Flags:, QOS:-1, S(nh):29,  Stats:9/882,  SPort 58271,
 TTL 0, Sinfo 4.0.0.0)
```

The statistics in the forward and reverse flow are incremented when traffic is flowing through. If statistics are not getting incremented for a particular flow, that can indicate a potential problem in that direction. The flow action should be `F` or `N` for the packets to be forwarded or NATed out. A flow action of `D` indicates that packets will be dropped.

7. Run the `vrouter_agent_debug` script to collect all of the relevant logs.

Inter-Compute Use Case

In an inter-compute case, the next-hop lookup points to the tunnel that takes the packet to the other compute node. The bridge entry will also indicate the Label/VNID added to the packet during encapsulation. Inter-compute traffic is between VMs on different compute nodes.

For Compute 1:

1. Discover the `vif` interfaces corresponding to the virtual machine interfaces (VMI)s of the VM by using the command:

   ```
   vif --list
   ```

   You can also discover the `vif` interfaces by entering the introspect URL:

   Example:

   ```
   http://10.1.1.5:8085/Snh_ItfReq?
   name=&type=&uuid=&vn=&mac=&ipv4_address=&ipv6_address=&parent_uuid=&ip_active=&ip6_active=&l2_
   active=
   ```

   NOTE: Replace the IP address with the actual compute IP address in the introspect HTTP URL.

2. Run the `vif --get <index>` command to verify the virtual routing and forwarding (VRF) and Policy flags are set in the vRouter interface (VIF).

Example output verifying flags for each `vif`:

```
vif0/4      OS: tapdd789d34-27
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.5
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:30  bytes:8676 errors:0
            TX packets:170  bytes:7140 errors:0
            ISID: 0 Bmac: 02:dd:78:9d:34:27
            Drops:81

vif0/6      OS: tapaedbc037-bf
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.6
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:96316  bytes:4858043 errors:0
            TX packets:96562  bytes:4884177 errors:0
            ISID: 0 Bmac: 02:ae:db:c0:37:bf
```

3. Run the following command to display all of the entries from the bridge table:

```
rt --dump <vrf id> --family bridge
```

Example:

```
rt --dump 2 --family bridge
Flags: L=Label Valid, Df=DHCP flood, Mm=Mac Moved, L2c=L2 Evpn Control Word, N=New Entry,
Ec=EvpnControlProcessing
vRouter bridge table 0/2
Index       DestMac                  Flags           Label/VNID      Nexthop           Stats
31264       0:0:5e:0:1:0             Df                   -               3          206834
79784       2:dd:78:9d:34:27         Df                   -              44               4
112924      ff:ff:ff:ff:ff:ff        LDf                  5              48              35
115240      2:0:0:0:0:2              Df                   -              12               0
169408      2:ae:db:c0:37:bf         Df                   -              25               1
183944      2:99:ef:64:96:e1         LDf                 27              20               0
205564      2:0:0:0:0:1              Df                   -              12               0
252380      0:25:90:c5:58:94         Df                   -               3               0
```

In the example, `2:99:ef:64:96:e1` belongs to IP address `10.1.1.7` and label `27` is used to encapsulate the packet.

4. Run `nh --get <nh id>` to get the next hop details.

Example:

```
nh --get 20
Id:20         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:12         Vrf:0
              Flags:Valid, MPLSoGRE, Etree Root,
              Oif:0 Len:14 Data:00 25 90 c5 62 1c 00 25 90 c5 58 94 08 00
              Sip:10.204.217.86 Dip:10.204.217.70
```

In the example, the next-hop output indicates the next-hop type as `Tunnel`, encapsulation used as `MPLSoGRE`, the outgoing interface as `Oif:0`, and the corresponding source and destination IP addresses of the tunnel.

For Compute 2:

1. Run the `mpls --get <label>` command to see the next hop mapped to the particular incoming MPLS table.

   Example:

```
mpls --get 27
MPLS Input Label Map

   Label    NextHop
-------------------------
      27        28
```

2. Run `nh --get <nh_id>` to get the next hop details.

   Example:

```
nh --get 28
Id:28         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:3          Vrf:2
              Flags:Valid, Policy, Etree Root,
              EncapFmly:0806 Oif:3 Len:14
              Encap Data: 02 99 ef 64 96 e1 00 00 5e 00 01 00 08 00
```

3. Run `vif --get <oifindex>` to get the outgoing VIF details.

Example:

```
vif --get 3

vif0/3      OS: tap99ef6496-e1
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.7
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:34  bytes:10044 errors:0
            TX packets:1699  bytes:78990 errors:0
            Drops:93
```

NOTE: If you are using VXLAN encapsulation, do the following on Compute 2:

a. For Step "1" on page 74 , instead of running the `mpls --get` command, run the `vxlan --get <vxlanid>` command to get the mapping from VXLAN ID to the next hop.

b. With VXLAN, the next hop points to a VRF translated next hop. Use the bridge lookup in the corresponding VRF, as shown in Step "3" on page 73 to get the final outgoing next hop, which will point to the VIF interface.

Unicast Packet Path - Inter-VN

IN THIS SECTION

- Intra-Compute Use Case | 76
- Inter-Compute Use Case | 79

The following procedure steps through debugging the packet path from VM1 to VM2 (on the same compute node) and VM3 (on a different compute node). In this example, the virtual machines (VMs) listed are in the same subnet `10.1.1.0/24`.

VM1 IP address `10.1.1.5/32` (Compute 1)

VM2 IP address `20.1.1.6/32` (Compute 1)

VM3 IP address `20.1.1.5/32` (Compute 2)

NOTE: Replace the IP address with the actual compute IP address in all of the introspect URLs.

Intra-Compute Use Case

1. Discover the `vif` interfaces corresponding to the virtual machine interfaces (VMI)s of the VM using the command:

```
vif --list
```

   You can also discover the `vif` interfaces by entering the introspect URL:

   Example:

```
http://10.1.1.5:8085/Snh_ItfReq?
name=&type=&uuid=&vn=&mac=&ipv4_address=&ipv6_address=&parent_uuid=&ip_active=&ip6_active=&l2_
active=
```

   NOTE: Replace the IP address with the actual compute IP address in the introspect HTTP URLs.

2. Run the `vif --get <index>` command to verify the virtual routing and forwarding (VRF) and Policy flags are set in the vRouter interface (VIF).

   Example output verifying flags for each `vif`:

```
vif0/4      OS: tapdd789d34-27
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.5
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:30  bytes:8676 errors:0
            TX packets:170  bytes:7140 errors:0
            ISID: 0 Bmac: 02:dd:78:9d:34:27
            Drops:81

vif0/6      OS: tapaedbc037-bf
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:10.1.1.6
            Vrf:2 Mcast Vrf:2 Flags:PL3L2Er QOS:-1 Ref:6
```

```
            RX packets:96316  bytes:4858043 errors:0
            TX packets:96562  bytes:4884177 errors:0
            ISID: 0 Bmac: 02:ae:db:c0:37:bf
            Drops:2
```

3. Run the following command to display all of the entries from the bridge table:

```
rt --dump <vrf id> --family bridge
```

Example:

```
rt --dump 2 --family bridge
Flags: L=Label Valid, Df=DHCP flood, Mm=Mac Moved, L2c=L2 Evpn Control Word, N=New Entry,
Ec=EvpnControlProcessing
vRouter bridge table 0/2
Index       DestMac                  Flags           Label/VNID      Nexthop          Stats
31264       0:0:5e:0:1:0             Df                   -              3         212744
79784       2:dd:78:9d:34:27         Df                   -             44            408
112924      ff:ff:ff:ff:ff:ff        LDf                  5             51             38
115240      2:0:0:0:0:2              Df                   -             12              0
169408      2:ae:db:c0:37:bf         Df                   -             25            405
183944      2:99:ef:64:96:e1         LDf                  5             37              0
205564      2:0:0:0:0:1              Df                   -             12              0
252380      0:25:90:c5:58:94         Df                   -              3              0
```

In the case of inter-virtual network (VN)s, the packets are Layer 3 routed instead of Layer 2 switched. The vRouter does a proxy ARP for the destination network providing it's virtual MAC address `0:0:5e:0:1:0` back for the ARP request from the source. This can be seen from the `rt -dump` of the source VN inet table. This results in the packet being received by the vRouter, which does the route lookup to send the packet to the correct destination.

4. Run `nh --get <nh id>` to display the next-hop details.

Example:

```
nh --get 3
Id:3          Type:L2 Receive     Fmly: AF_INET  Rid:0  Ref_cnt:8          Vrf:0
              Flags:Valid, Etree Root,
```

5. Run `rt --dump 2 --family inet | grep <ip address>` to display inet family routes on the specified IP address.

Example:

```
rt --dump 2 --family inet | grep 20.1.1.6
Destination           PPL        Flags        Label         Nexthop    Stitched MAC(Index)

20.1.1.6/32            32            P          -             30        -
20.1.1.60/32            0                       -              0        -
20.1.1.61/32            0                       -              0        -
20.1.1.62/32            0                       -              0        -
20.1.1.63/32            0                       -              0        -
20.1.1.64/32            0                       -              0        -
20.1.1.65/32            0                       -              0        -
20.1.1.66/32            0                       -              0        -
20.1.1.67/32            0                       -              0        -
20.1.1.68/32            0                       -              0        -
20.1.1.69/32            0                       -              0        -
20.1.106.0/24           0                       -              0        -
```

6. Run `nh --get <nh id>` to get the next hop details.

   Example:

```
nh --get 30
Id:30      Type:Encap          Fmly: AF_INET  Rid:0  Ref_cnt:5          Vrf:3
              Flags:Valid, Policy, Etree Root,
              EncapFmly:0806 Oif:3 Len:14
              Encap Data: 02 60 fc 55 cb bf 00 00 5e 00 01 00 08 00
```

7. Run `vif --get <oifindex>` to get the outgoing VIF details.

   Example:

```
vif --get 3
vif0/3      OS: tap60fc55cb-bf
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:20.1.1.6
            Vrf:3 Mcast Vrf:3 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:356  bytes:32586 errors:0
            TX packets:3930  bytes:177290 errors:0
            ISID: 0 Bmac: 02:60:fc:55:cb:bf
            Drops:84
```

Inter-Compute Use Case

In the case of inter-compute, the next hop looked up to send the packet out, will point to a tunnel next hop. Depending on the encapsulation priority, the appropriate encapsulation is added and the packet is tunneled out. Inter-compute traffic is between VMs on different compute nodes.

For Compute 1:

1. Run `rt --dump 2 --family inet | grep <ip address>` to display inet family routes for a specified IP address.

   Example:

```
rt --dump 2 --family inet | grep 20.1.1.5
20.1.1.5/32            32           LP       32             49        -
20.1.1.50/32            0                        -              0        -
20.1.1.51/32            0                        -              0        -
20.1.1.52/32            0                        -              0        -
20.1.1.53/32            0                        -              0        -
20.1.1.54/32            0                        -              0        -
20.1.1.55/32            0                        -              0        -
20.1.1.56/32            0                        -              0        -
20.1.1.57/32            0                        -              0        -
20.1.1.58/32            0                        -              0        -
```

2. Run `nh --get <nh id>` to display the next-hop details, which points to a tunnel next hop.

   Example:

```
nh --get 49
Id:49      Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:9          Vrf:0
              Flags:Valid, MPLSoUDP, Etree Root,
              Oif:0 Len:14 Data:00 25 90 c5 62 1c 00 25 90 c5 58 94 08 00
              Sip:10.204.217.86 Dip:10.204.217.70
```

For Compute 2:

1. Run the `mpls --get <label>` command to see the next hop mapped to the particular incoming MPLS table.

Example:

```
mpls --get 32
MPLS Input Label Map

   Label    NextHop
------------------------
      32        36
```

2. Run `nh --get <nh id>` to view the next hop details.

   Example:

```
nh --get 36
Id:36      Type:Encap          Fmly: AF_INET  Rid:0  Ref_cnt:5          Vrf:3
              Flags:Valid, Policy, Etree Root,
              EncapFmly:0806 Oif:4 Len:14
              Encap Data: 02 f3 37 7b 53 25 00 00 5e 00 01 00 08 00
```

   In the example, `Oif:4` is the OIF index in the next hop which is the outgoing interface for the packet. The `Encap Data` corresponds to the L2 encapsulation that is added to the IP packet before the packet is forwarded to the outgoing interface.

3. Run `vif --get <oifindex>` to get the outgoing VIF details.

   Example:

```
vif --get 4
vif0/4      OS: tapf3377b53-25
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:20.1.1.5
            Vrf:3 Mcast Vrf:3 Flags:PL3L2Er QOS:-1 Ref:6
            RX packets:100  bytes:16915 errors:0
            TX packets:3955  bytes:178363 errors:0
            ISID: 0 Bmac: 02:f3:37:7b:53:25
            Drops:78
```

   For details about EVPN type 5 routing in Contrail Networking, see Support for EVPN Route Type 5.

Broadcast, Unknown Unicast, and Multicast Packet Path

The following procedure steps through debugging the packet path for broadcast, unknown unicast, and multicast (BUM) traffic in Contrail Networking. In this example, the virtual machines (VMs) listed are in the same subnet 70.70.70.0/24.

The ToR Service Node (TSN) actively holds the `contrail-tor-agent` and is responsible for:

1. Acting as a receiver of all BUM traffic coming from the ToR switch.
2. Acting as DNS/DHCP responder for the BMS connected to the ToR switch.

Contrail Networking releases earlier than 5.x, used an Open vSwitch Database (OVSDB)-managed VXLAN environment.

Topology example for an OVSDB-managed VXLAN:

- Top-of-Rack Switch 1 (ToR SW1) - 10.204.74.229 (lo0.0 = 1.1.1.229)
- Top-of-Rack Switch 2 (ToR SW2) - 10.204.74.230 (lo0.0 = 1.1.1.230)
- ToR Services Node 1 (TSN1) = 10.219.94.7
- ToR Services Node 2 (TSN2) = 10.219.94.8
- Controller1 = 10.219.94.4
- Controller2 = 10.219.94.5
- Controller3 = 10.219.94.6
- Compute1 = 10.219.94.9
- Compute2 = 19.219.94.18
- Virtual Network (VN) = 70.70.70.0/24
- Virtual Machine 1 (VM1) = 70.70.70.3 residing on Compute2
- Virtual Machine 2 (VM2) = 70.70.70.5 residing on Compute1
- Bare Metal Server 1 (BMS1) = 70.70.70.100
- Bare Metal Server 2 (BMS2) = 70.70.70.101

1. Run the `set protocols ovsdb interfaces <interface>` command to configure the physical interfaces that you want the OVSDB protocol to manage.

Example:

```
set protocols ovsdb interfaces ge-0/0/46
set protocols ovsdb interfaces ge-0/0/90
```

The ToR interfaces from which the BMS hangs are marked as `ovsdb interfaces`.

2. View packets coming into these interfaces by displaying the `ovsdb mac` table for the ToR switch.

   Example:

```
root@QFX5100-229> show ovsdb mac
Logical Switch Name: Contrail-9ed1f70a-6eac-4cdb-837a-1579728fd9a1
  Mac                 IP              Encapsulation     Vtep
  Address             Address                           Address
  ff:ff:ff:ff:ff:ff   0.0.0.0         Vxlan over Ipv4   1.1.1.229
  40:a6:77:d8:37:1d   0.0.0.0         Vxlan over Ipv4   1.1.1.229
  02:3b:ce:56:61:98   0.0.0.0         Vxlan over Ipv4   10.219.94.18
  02:53:89:c4:29:1c   0.0.0.0         Vxlan over Ipv4   10.219.94.18
  02:72:e9:7a:cd:f5   0.0.0.0         Vxlan over Ipv4   10.219.94.9
  02:75:a1:33:65:3c   0.0.0.0         Vxlan over Ipv4   10.219.94.9
  ff:ff:ff:ff:ff:ff   0.0.0.0         Vxlan over Ipv4   10.219.94.7

{master:0}
```

The entry marked in red (`ff:ff:ff:ff:ff:ff` - broadcast route) indicates the next hop for a BUM packet coming into the ToR SW's `ovsdb interface`. In this case, VTEP address `10.219.94.7` is the next hop, which is TSN1. This changes based on which TSN has the active `contrail-tor-agent` for the ToR switch in question. With this, the BUM packet is forwarded to the TSN node in a VXLAN tunnel (local VTEP source interface is `1.1.1.229` and RVTEP source interface is `10.219.94.7`).

The VXLAN encapsulated packet is sent with a VXLAN Network Identifier (VNI) that is predetermined by Contrail Networking when logical interfaces are created. For example, when `ge-0/0/46` was configured as a logical port in Contrail Networking, the following configuration was committed on the ToR.

Example:

```
set vlans Contrail-9ed1f70a-6eac-4cdb-837a-1579728fd9a1 interface ge-0/0/46.0
set vlans Contrail-9ed1f70a-6eac-4cdb-837a-1579728fd9a1 interface ge-0/0/90.0
set vlans Contrail-9ed1f70a-6eac-4cdb-837a-1579728fd9a1 vxlan vni 4
```

As the VXLAN encapsulated packet arrives on the TSN node, let's examine how the vRouter handles this packet.

3. Run `vxlan --dump` to dump the VXLAN table. The VXLAN table maps a network ID to a next hop.

   Example:

```
root@contrail61:~# vxlan --dump
VXLAN Table

 VNID    NextHop
----------------
      4    13
```

   In the example, next hop `13` is programmed for VNI `4`.

4. Run `nh --get <nh id>` to display the next-hop details and determine the virtual routing and forwarding (VRF) associated.

   Example:

```
root@contrail61:~# nh --get 13
Id:13         Type:Vrf_Translate  Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Vxlan,
              Vrf:1
```

5. Run the following command to display all of the entries from the bridge table:

```
rt --dump <vrf id> --family bridge
```

   Example:

```
root@contrail61:~# rt --dump 1 --family bridge
Flags: L=Label Valid, Df=DHCP flood
vRouter bridge table 0/1
Index       DestMac                  Flags           Label/VNID      Nexthop
30780       0:1a:a0:e:30:26                               -             1
57812       2:53:89:c4:29:1c          LDf                17            15
70532       2:3b:ce:56:61:98          LDf                20            15
87024       2:72:e9:7a:cd:f5          LDf                17            14
97192       ff:ff:ff:ff:ff:ff         LDf                 4            24
```

```
121160      0:1a:a0:a:b4:87                          -            1
225832      40:a6:77:d8:37:1d        LDf             4           19
237084      0:11:a:6c:50:d4           Df             -            3
244992      aa:bb:cc:dd:3e:f4                        -            1
252916      0:0:5e:0:1:0              Df             -            3
256476      2:75:a1:33:65:3c         LDf            20           14
```

In the example bridge table, since we are tracing the BUM packet path, we need to examine the ff:ff:ff:ff:ff:ff route by selecting the next hop programmed. In the example, it is 24. Note that a series of composite next hops are programmed.

6. Run `nh --get <nh id>` to display the next-hop details.

   Example:

```
root@contrail61:~# nh --get 24
Id:24         Type:Composite      Fmly:AF_BRIDGE  Rid:0  Ref_cnt:4          Vrf:1
              Flags:Valid, Multicast, L2,
              Sub NH(label): 20(0) 22(0) 21(0)

Id:20         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Tor,
              Sub NH(label): 19(4)

Id:19         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:3          Vrf:0
              Flags:Valid, Vxlan,
              Oif:0 Len:14 Flags Valid, Vxlan,  Data:00 00 5e 00 01 21 00 11 0a 6c 50 d4 08
00
              Vrf:0  Sip:10.219.94.7  Dip:1.1.1.229  << Source where the BUM Traffic came
from.

Id:22         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Evpn,
              Sub NH(label):

Id:21         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Fabric,
              Sub NH(label): 15(4099)

Id:15         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:6          Vrf:0
              Flags:Valid, MPLSoGRE,
              Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:f8 bc 12 33 43 31 00 11 0a 6c 50 d4
08 00
```

```
                Vrf:0  Sip:10.219.94.7  Dip:10.219.94.18  << Compute node which has a VM in
this VN.
```

The multicast tree in the example shows that there are two Dynamic IPs (DIP)s. The DIP where the packet came from is ignored. Therefore, packet gets forwarded to DIP `10.219.94.18` only.

7. Run `vxlan --get <vnid>` to examine what DIP `10.219.94.18` does with the incoming VXLAN encapsulated packet.

   Example:

```
root@contrail4:~# vxlan --get 4
VXLAN Table

 VNID    NextHop
----------------
      4    20
```

8. Run `nh --get <nh id>` to display the next-hop details.

   Example:

```
root@contrail4:~# nh --get 20
Id:20         Type:Vrf_Translate  Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Vxlan,
              Vrf:1
```

9. Run the following command to display all of the entries from the bridge table:

```
rt --dump <vrf id> --family bridge
```

   Example:

```
root@contrail4:~# rt --dump 1 --family bridge
Flags: L=Label Valid, Df=DHCP flood
vRouter bridge table 0/1
Index       DestMac                  Flags           Label/VNID      Nexthop
57812       2:53:89:c4:29:1c                              -           15
70532       2:3b:ce:56:61:98                              -           22
87024       2:72:e9:7a:cd:f5          LDf                17           25
```

```
97192       ff:ff:ff:ff:ff:ff         LDf                4            50
112856      f8:bc:12:33:43:31          Df                -             3
225832      40:a6:77:d8:37:1d         LDf                4            18
252916      0:0:5e:0:1:0               Df                -             3
256476      2:75:a1:33:65:3c          LDf               20            25
```

In the example bridge table, since we are tracing the BUM packet path, we need to examine the ff:ff:ff:ff:ff:ff route by selecting the next hop programmed. In the example, it is 50.

10. Run `nh --get <nh id>` to display the next-hop details.

Example:

```
root@contrail4:~# nh --get 50
Id:50         Type:Composite      Fmly:AF_BRIDGE  Rid:0  Ref_cnt:4          Vrf:1
              Flags:Valid, Multicast, L2,
              Sub NH(label): 43(0) 49(0)

Id:43         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Fabric,
              Sub NH(label): 31(4612) 25(4617)

Id:31         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:0
              Flags:Valid, MPLSoGRE,
              Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:00 11 0a 6c 50 d4 f8 bc 12 33 43 31
08 00
              Vrf:0  Sip:10.219.94.18  Dip:10.219.94.7  <<< Source where the BUM traffic
came from.

Id:25         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:2562       Vrf:0
              Flags:Valid, MPLSoGRE,
              Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:44 a8 42 3a 94 f4 f8 bc 12 33 43 31
08 00
              Vrf:0  Sip:10.219.94.18  Dip:10.219.94.9  <<< Compute node which has a VM in
this VN.

Id:49         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Encap,
              Sub NH(label): 14(0) 21(0)

Id:14         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid,
              EncapFmly:0806 Oif:4 Len:14
```

```
              Encap Data: 02 53 89 c4 29 1c 00 00 5e 00 01 00 08 00

Id:21         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid,
              EncapFmly:0806 Oif:3 Len:14
              Encap Data: 02 3b ce 56 61 98 00 00 5e 00 01 00 08 00  <<< Local VM belonging
to this VN that is an intended receiver of this multicast traffic.
```

In the example, you only have to inspect DIP `10.219.94.9`. The remaining endpoints are either local or the source where the BUM traffic came from. Now, let us examine, what DIP `10.219.94.9` does with the incoming VXLAN encapsulated packet.

11. Run `vxlan --get <vnid>` to examine what DIP `10.219.94.9` does with the incoming VXLAN encapsulated packet.

    Example:

```
root@contrail101:~# vxlan --get 4
VXLAN Table

 VNID    NextHop
----------------
      4    20
```

12. Run `nh --get <nh id>` to display the next-hop details.

    Example:

```
root@contrail101:~# nh --get 20
Id:20         Type:Vrf_Translate  Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Vxlan,
              Vrf:1
```

13. Display the bridge table for the VRF by using the following command:

```
rt --dump <vrf id> --family bridge
```

Example:

```
root@contrail101:~# rt --dump 1 --family bridge
Flags: L=Label Valid, Df=DHCP flood
vRouter bridge table 0/1
Index       DestMac                  Flags           Label/VNID      Nexthop
57812       2:53:89:c4:29:1c          LDf                17            28
70532       2:3b:ce:56:61:98          LDf                20            28
87024       2:72:e9:7a:cd:f5                              -            15
97192       ff:ff:ff:ff:ff:ff         LDf                4            31
140744      44:a8:42:3a:94:f4          Df                -             3
225832      40:a6:77:d8:37:1d         LDf                4            24
252916      0:0:5e:0:1:0               Df                -             3
256476      2:75:a1:33:65:3c                             -            22
              Encap Data: f8 bc 12 33 43 31 44 a8 42 3a 94 f4 08 00
```

14. Run `nh --get <nh id>` to display the next-hop details.

Example:

```
root@contrail101:~# nh --get 31
Id:31         Type:Composite      Fmly:AF_BRIDGE  Rid:0  Ref_cnt:4          Vrf:1
              Flags:Valid, Multicast, L2,
              Sub NH(label): 30(0) 36(0)

Id:30         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Fabric,
              Sub NH(label): 29(4612) 28(4099)

Id:29         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:0
              Flags:Valid, MPLSoGRE,
              Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:00 11 0a 6c 50 ac 44 a8 42 3a 94 f4
08 00
              Vrf:0  Sip:10.219.94.9  Dip:10.219.94.8 << TSN2 in this topology that is
managing a ToR with an end-point belonging to this VN.

Id:28         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:2566       Vrf:0
              Flags:Valid, MPLSoGRE,
              Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:f8 bc 12 33 43 31 44 a8 42 3a 94 f4
08 00
              Vrf:0  Sip:10.219.94.9  Dip:10.219.94.18 << Source where the BUM traffic came
```

```
from.

Id:36         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Encap,
              Sub NH(label): 14(0) 21(0)

Id:14         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid,
              EncapFmly:0806 Oif:3 Len:14
              Encap Data: 02 72 e9 7a cd f5 00 00 5e 00 01 00 08 00 << local VM that is an
intended receiver of this traffic as it is tagged to this VN

Id:21         Type:Encap          Fmly:AF_BRIDGE  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid,
              EncapFmly:0806 Oif:4 Len:14
              Encap Data: 02 75 a1 33 65 3c 00 00 5e 00 01 00 08 00  << Local VM that is an
intended receiver of this traffic since it is tagged to this VN.
```

From the above output, the only DIP that you have to further examine is `10.219.94.8`. The remaining DIPs are either local or the source where the BUM traffic came from. Now, let's examine what DIP `10.219.94.8` does with the incoming VXLAN encapsulated packet.

15. Run `vxlan --get <vnid>` to examine what DIP `10.219.94.9` does with the incoming VXLAN encapsulated packet.

    Example:

```
root@contrail66:~# vxlan --get 4
VXLAN Table

 VNID    NextHop
----------------
      4    14
```

16. Run `nh --get <nh id>` to display the next-hop details.

    Example:

```
root@contrail66:~# nh --get 14
Id:14         Type:Vrf_Translate  Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
```

```
                Flags:Valid, Vxlan,
                Vrf:1
```

17. Display the bridge table for the VRF by using the following command:

```
rt --dump <vrf id> --family bridge
```

Example:

```
root@contrail66:~# rt --dump 1 --family bridge
Flags: L=Label Valid, Df=DHCP flood
vRouter bridge table 0/1
Index       DestMac                   Flags           Label/VNID      Nexthop
30780       0:1a:a0:e:30:26                                 -           1
57812       2:53:89:c4:29:1c           LDf                 17          17
70532       2:3b:ce:56:61:98           LDf                 20          17
87024       2:72:e9:7a:cd:f5           LDf                 17          16
97192       ff:ff:ff:ff:ff:ff          LDf                  4          24
121160      0:1a:a0:a:b4:87                                 -           1
217208      0:11:a:6c:50:ac             Df                  -           3
225832      40:a6:77:d8:37:1d          LDf                  4          20
244992      aa:bb:cc:dd:3e:f4                               -           1
252916      0:0:5e:0:1:0                Df                  -           3
256476      2:75:a1:33:65:3c           LDf                 20          16
```

18. Run `nh --get <nh id>` to display the next-hop details.

Example:

```
root@contrail66:~# nh --get 24
Id:24         Type:Composite      Fmly:AF_BRIDGE  Rid:0  Ref_cnt:4          Vrf:1
              Flags:Valid, Multicast, L2,
              Sub NH(label): 23(0) 25(0) 21(0)

Id:23         Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
              Flags:Valid, Tor,
              Sub NH(label): 22(4)

Id:22         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:0
              Flags:Valid, Vxlan,
```

```
                Oif:0 Len:14 Flags Valid, Vxlan,  Data:00 00 5e 00 01 21 00 11 0a 6c 50 ac 08
00
                Vrf:0  Sip:10.219.94.8  Dip:1.1.1.230 <<< Another ToR switch that has an end-
point belonging to this VN.

Id:25           Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
                Flags:Valid, Evpn,
                Sub NH(label):

Id:21           Type:Composite      Fmly: AF_INET  Rid:0  Ref_cnt:2          Vrf:1
                Flags:Valid, Fabric,
                Sub NH(label): 16(4617)

Id:16           Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:6          Vrf:0
                Flags:Valid, MPLSoGRE,
                Oif:0 Len:14 Flags Valid, MPLSoGRE,  Data:44 a8 42 3a 94 f4 00 11 0a 6c 50 ac
08 00
                Vrf:0  Sip:10.219.94.8  Dip:10.219.94.9 <<< Source where the BUM traffic came
from.
```

Now, you just have one DIP `1.1.1.230` which is the ToR SW2 in the topology. This should also be present in the multicast tree as this ToR SW also has an end-point (which is BMS2) in the same VN (VNI=4) as the one we are tracing.

This completes all levels of forwarding and tracing the BUM packet from one ToR switch and is replicated to other intended receivers in the topology.

These multicast trees are programmed by the controllers that the TSN is connected to. If you want to inspect the controller's memory and what eventually gets programmed on all TSN computes, enter the following introspect URL using your controller IP address:

```
http://<controller_ip>:8083/Snh_ShowMulticastManagerDetailReq?x=default-domain:admin:seventy-
network:seventy-network.ermvpn.0
```

RELATED DOCUMENTATION

Assisted Replication of Broadcast, Unknown Unicast, and Multicast Traffic

Using Contrail Tools

Contrail-tools container provides centralized location for all the available tools and CLI commands in one place.

Starting with Contrail Networking Release 2008, `contrail-tools` command will be installed by default.

`contrail-tools` command enables you to log in to the contrail-tools container and execute the tool. Additionally, the command will kill the container on exit.

Table 5 on page 92 provides a list of available tools and CLI options in the *contrail-tools* package.

Table 5: Available Tools and CLI options

Tools and CLI commands	Description
dpdkinfo	Adds support to display bond, lacp, Nic, mempool, core, and app information.
dpdkvifstats.py	Display the PPS statistics of DPDK vRouter.
dropstats	Inspects packet drop counters in the vRouter.
flow	Displays active flows in the system.
mirror	Displays the mirror table entries.
mpls	Displays the input label map programmed into the vRouter.
nh	Displays the next hops that the vRouter knows.
qosmap	Retrieves and sets QoS mappings.
rt	Displays routes in virtual routing and forwarding (VRF).
sandump	Captures the Sandesh messages from the netlink connection between Agent and vRouter.
vif	Inspects vRouter interfaces associated with the vRouter module.

Table 5: Available Tools and CLI options *(Continued)*

Tools and CLI commands	Description
vifdump	Captures and analyzes packets from DPDK interface.
vrfstats	Displays the next hop statistics for the VRF.
vrftable	Displays the interface mapping for each VRF for a host-based firewall feature.
vrinfo	Displays internal state of DPDK/Kernel vRouter.
vrmemstats	Displays the vRouter memory usage statistics.
vrouter	Display the vRouter information.
vxlan	Displays the vxlan table entries.

There are 2 ways to execute the `contrail-tools` command:

- Execute `contrail-tools` command to login to the container.

  For example:

```
[root]# contrail-tools
(contrail-tools)[root /]$ vif
Usage: vif [--create <intf_name> --mac <mac>]
   [--add <intf_name> --mac <mac> --vrf <vrf>
   --type [vhost|agent|physical|virtual|monitoring]
   --transport [eth|pmd|virtual|socket]
   --xconnect <physical interface name>
   --policy, --vhost-phys, --dhcp-enable]
   --vif <vif ID> --id <intf_id> --pmd --pci]
   [--delete <intf_id>|<intf_name>]
   [--get <intf_id>][--kernel][--core <core number>][--rate] [--get-drop-stats]
   [--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
   [--list][--core <core number>][--rate]
   [--sock-dir <sock dir>]
   [--help]
```

- Execute `contrail-tools` command with the CLI as argument.

  For example:

```
[root]# contrail-tools vif
Usage: vif [--create <intf_name> --mac <mac>]
   [--add <intf_name> --mac <mac> --vrf <vrf>
   --type [vhost|agent|physical|virtual|monitoring]
   --transport [eth|pmd|virtual|socket]
   --xconnect <physical interface name>
   --policy, --vhost-phys, --dhcp-enable]
   --vif <vif ID> --id <intf_id> --pmd --pci]
   [--delete <intf_id>|<intf_name>]
   [--get <intf_id>][--kernel][--core <core number>][--rate] [--get-drop-stats]
   [--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
   [--list][--core <core number>][--rate]
   [--sock-dir <sock dir>]
   [--help]
```

Change History Table

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

Release	Description
2008	Starting with Contrail Networking Release 2008, `contrail-tools` command will be installed by default.

RELATED DOCUMENTATION

Using Sandump Tool | 94

Using Sandump Tool

Starting with Contrail Networking Release 2008, *Sandump* tool is available in contrail-tools container. You can use the *Sandump* tool on macOS machines.

Sandump tool captures the Sandesh messages from netlink connection between Agent and vRouter (only DPDK mode) and provides interpretation of all the captured bytes.

Starting with Contrail Networking Release 2011, you can use *Sandump* tool on Windows machines.

Sandesh is 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 Insights node.

You can analyze the captured bytes in Wireshark. The Wireshark plugin parses the hex dumps of all Sandesh objects. You must use Wireshark Release 3.2 and later.

You must have Wireshark application installed on your machine. You can download Wireshark from the Download Wireshark page.

For more details on Wireshark, see https://www.wireshark.org/docs/.

Follow the procedure to use Sandump tool:

1. Run the `sandump` command. It gives summary of each message which is being transferred between the agent and the vRouter.

```
 (vrouter-agent-dpdk)[root]$ ./sandump -h
Sandump - Sandesh dump utility
Usage:
       ./sandump -w <filename> [filename to write the sandesh packets]
       ./sandump -c <filename> [force cleanup]
(vrouter-agent-dpdk)[root]$
```

2. Copy the output into a file.

```
(vrouter-agent-dpdk)[root]$ ./sandump -w <filename>.pcap
Dumping into <filename>.pcap
Running as user "root" and group "root". This could be dangerous.
Capturing on 'lo'
12 ^C
./sandump: closing...
(vrouter-agent-dpdk)[root]$
```

   The command generates a file which contains sniffed bytes converted in to the pcap format.

3. Analyze the captured packets transferred between the agent and the vRouter.

```
(vrouter-agent-dpdk)[root]$ ./sandump
Running as user "root" and group "root". This could be dangerous.
Capturing on 'lo'
    1 2020-08-04 09:51:01.233639252        Agent → Vrouter      Vif 790  Operation: Dump
Type: Host  ID: 0
```

```
    2 2020-08-04 09:51:01.251279611      Vrouter → Agent        Response, Vif 3966  Response:
0x0000000, Multiple  vr_interface_req
    3 2020-08-04 09:51:33.290323560        Agent → Vrouter      Mem Stats 869  Operation: Get
    4 2020-08-04 09:51:33.290964111      Vrouter → Agent        Response, Mem Stats 899
Response: 0x00000000
    5 2020-08-04 09:51:46.175797696        Agent → Vrouter      Info 137  ID: 0 Operation:
Dump
    6 2020-08-04 09:51:46.176494123      Vrouter → Agent        Response, Info 1949
Response: 0x00000001  ID: 0
    7 2020-08-04 09:51:58.920197081        Agent → Vrouter      Nexthop 280   Nexthop ID: 0
Operation: Dump
    8 2020-08-04 09:51:58.920905495      Vrouter → Agent        Response, Nexthop 3898
Response: 0x4000001, Multiple  vr_nexthop_req
    9 2020-08-04 09:51:58.922297667        Agent → Vrouter      Nexthop 280   Nexthop ID: 0
Operation: Dump
   10 2020-08-04 09:51:58.922425514      Vrouter → Agent        Response, Nexthop 3930
Response: 0x4000001, Multiple  vr_nexthop_req
   11 2020-08-04 09:51:58.923525453        Agent → Vrouter      Nexthop 280   Nexthop ID: 0
Operation: Dump
   12 2020-08-04 09:51:58.926925821      Vrouter → Agent        Response, Nexthop 792
Response: 0x0000000, Multiple  vr_nexthop_req
^C12 packets captured
./sandump: closing...
(vrouter-agent-dpdk)[root]$
```

4. Analyze the pcap file in WireShark.

- Follow the procedure to analyze the packets in Wireshark for Windows OS.

  a. Download the **sandump_wireshark_plugin** folder from the https://github.com/tungstenfabric/tf-vrouter/tree/master/utils/sandump repository.

  b. Copy the **sandump_wireshark_plugin/main.lua** file in **C:\Program Files\Wireshark\plugins** folder.

     Create new **lua** folder in **C:\Program Files\Wireshark** and copy the rest of the lua files present in **sandump_wireshark_plugin** folder to the newly created **lua** folder.

     NOTE: Wireshark installation directory for 32-bit Windows is present in **C:\Program Files (x86)\Wireshark**and for 64-bit Windows is present in **C:\Program Files \Wireshark**.

  c. Run Notepad as administrator and open **C:/Windows/System32/drivers/etc/hosts** file.

d. Add the host names with the following details:

- Agent IP address—0.0.0.0
- vRouter IP address—1.1.1.1

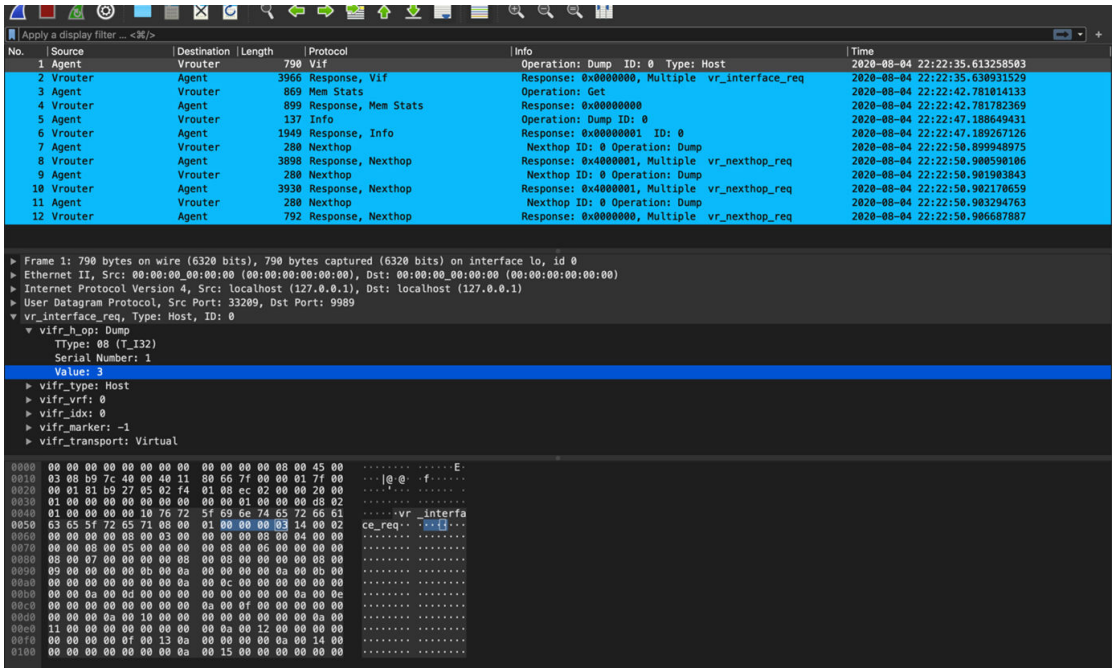
Figure 10 on page 97 shows the host file with the required IP addresses.

Figure 10: host file

```
# BEGIN hosts added by Pulse
  x.x.x.x example.net
# END hosts added by Pulse
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
0.0.0.0         Agent
1.1.1.1         Vrouter
~
~
~
```

e. Open the pcap file generated from Sandump tool for further debugging in Wireshark.

Figure 11: File debugging in Wireshark



- Follow the procedure to analyze the packets in Wireshark for macOS.

  a. Download the **sandump_wireshark_plugin** folder from the https://github.com/tungstenfabric/tf-vrouter/tree/master/utils/sandump repository.

  b. Copy the **sandump_wireshark_plugin** folder in **/Applications/Wireshark.app/Contents/PlugIns/wireshark** directory which is also know as *Global Lua Plugins* directory.

  c. Un-comment the `package.prepend_path(...)` line in main.lua, common.lua and helpers.lua files found in **sandump_wireshark_plugin** folder.

  d. Navigate to **Wireshark** > **About Wireshark** > **Folders** > **Personal configuration** to edit the configuration.

  e. Create hosts file in the **Personal configuration** directory and add the host names with the following details:

     - Agent IP address—0.0.0.0
     - vRouter IP address—1.1.1.1

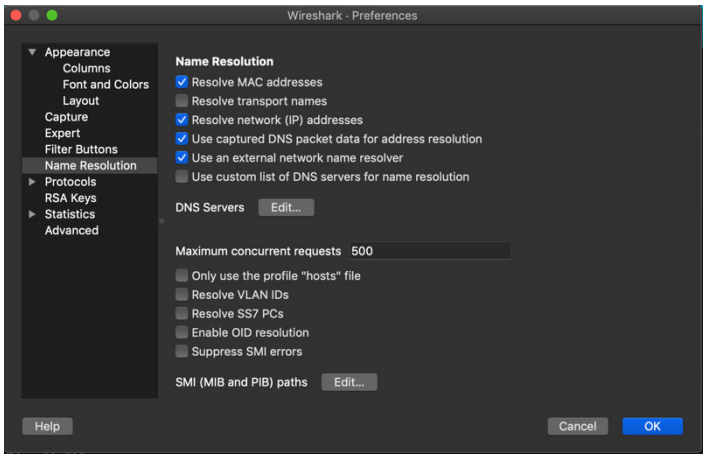
     Figure 12 on page 99 shows the host file with the required IP addresses.

Figure 12: host file

```
# BEGIN hosts added by Pulse
  x.x.x.x example.net
# END hosts added by Pulse
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
0.0.0.0         Agent
1.1.1.1         Vrouter
~
~
~
```

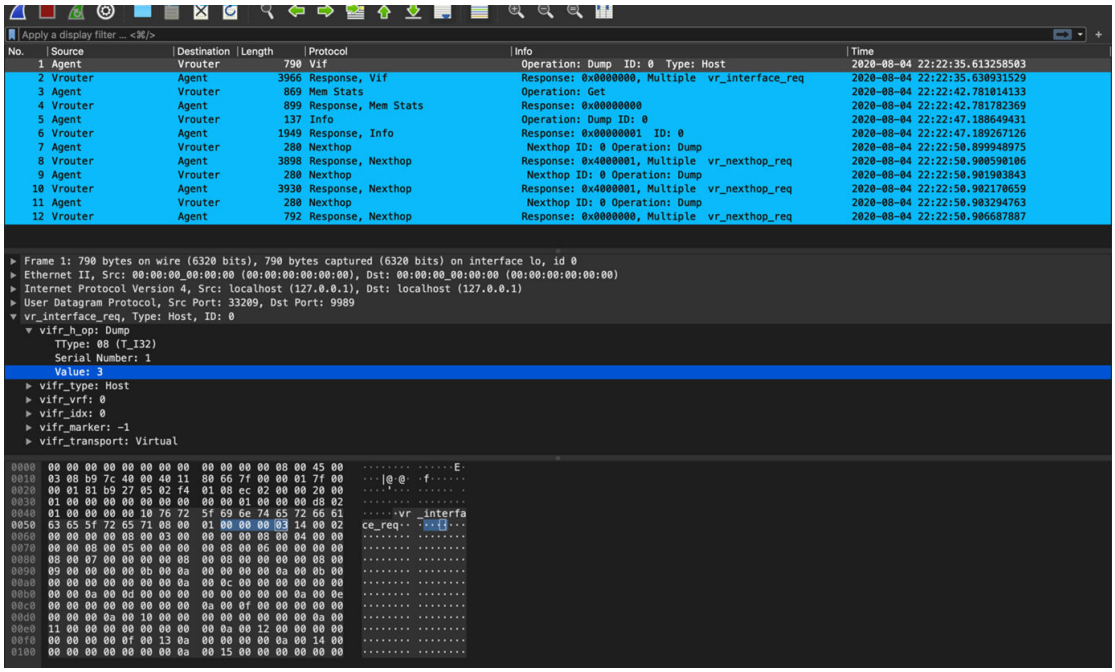
f. Navigate to **Wireshark > Preferences > Name Resolution** and check **Resolve network (IP) addresses** option.

Figure 13: Wireshark—Preferences



g. Open the pcap file generated from Sandump tool for further debugging in Wireshark.

Figure 14: File debugging in Wireshark



Change History Table

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

Release	Description
2011	Starting with Contrail Networking Release 2011, you can use *Sandump* tool on Windows machines.
2008	Starting with Contrail Networking Release 2008, *Sandump* tool is available in contrail-tools container. You can use the *Sandump* tool on macOS machines.

RELATED DOCUMENTATION

Using Contrail Tools | 92

Security Logging Object

IN THIS SECTION

- Defining an SLO | 101
- Attaching an SLO to a Virtual Network and Virtual Machine Interface | 102
- Editing an Existing SLO | 104

You can define a security logging object (SLO) to log sessions that match a specific policy rule or security group. An SLO also enables selective session logging. This reduces the amount of data sent from vRouter agent to Contrail Analytics.

You can attach an SLO to a:

- Virtual network
- Virtual machine interface

These topics provide information on how you can define an SLO, attach an SLO to a virtual network and virtual machine interface, associate a policy rule or security group to SLO, and edit the name of an existing SLO.

Defining an SLO

Follow these steps to define an SLO by using the Contrail Command user interface (UI).

These steps also describe how you can associate a network policy rule or security group to an SLO.

1. Navigate to **Security**>**Security Logging Object**.
   The Security Logging Object page is displayed.
2. Click **Create** to define a new security logging object.
3. Enter the following information in the Create Security Logging Object page.
   a. Enter a name for the SLO in the Name field.
   b. Enter the number of sessions logged in the Rate field.
      Rate indicates the number of sessions logged. The first session in every R (rate) number of sessions matching the SLO is logged. When the rate is set to 1, all sessions are logged.
   c. Select **Up** from the Admin State list to indicate the admin state of the security logging object.
   d. Select the network policy you want to attach to the SLO from the Network Policies list.

This enables logging of sessions for all virtual network interfaces that the selected network policy is attached to.

e. Select the security groups you want to attach to the SLO from the Security Group list.

This enables logging of sessions for all virtual machine interfaces that the selected security group is attached to.

f. You can also define a new SLO rule for a network policy and security group from the Rules section of the Create Security Logging Object page.

To define an SLO rule for a network policy,

i. Select **Network Policy** from the Type list.

ii. Select the network policy you want this SLO rule to be applied to, from the Network Policy list.

iii. Enter the number of sessions you want logged in the Rate field.

iv. To add another rule, click **+Add**.

To define an SLO rule for a security group,

a. Select **Security Group** from the Type list.

b. Select the security group you want this SLO rule to be applied to, from the Security Groups list.

c. Enter the number of sessions you want logged in the Rate field.

d. To add another rule, click **+Add**.

4. Click **Create** to create the SLO.

The Security Logging Object page is displayed.

Attaching an SLO to a Virtual Network and Virtual Machine Interface

IN THIS SECTION

- Attaching an SLO to a Virtual Network | **103**
- Attaching an SLO to a Virtual Machine Interface | **103**

After you have defined an SLO, you can attach the SLO to a virtual network and a virtual machine interface.

Follow these steps to attach an SLO to a virtual machine and a virtual machine interface.

Attaching an SLO to a Virtual Network

You can attach an SLO to a virtual network while creating the virtual network or after you have created the virtual network˙.

For steps to attach an SLO while creating a virtual network, see *Create Virtual Network*.

Follow these steps to attach an SLO to an existing virtual network.

1. Navigate to **Overlay**>**Virtual Networks**.
   The All networks page is displayed.
2. Select the virtual network you want to edit by click the **Edit** icon at the end of the row.
   The Edit Virtual Network page is displayed.
3. Click the **Advanced** section.
4. Select the SLO from the Security Logging Object list.
5. Click **Save** to save configuration.

Attaching an SLO to a Virtual Machine Interface

IN THIS SECTION

- Attaching an SLO to a Virtual Machine Interface while creating a Virtual Port | **103**
- Attaching an SLO to an existing Virtual Machine Interface | **104**

You can attach an SLO to a virtual machine interface while creating a virtual port or after you have created the virtual port.

Attaching an SLO to a Virtual Machine Interface while creating a Virtual Port

Follow these steps to attach an SLO to a virtual machine interface while creating a virtual port.

1. Navigate to **Overlay**>**Virtual Ports**.
   The Virtual Ports page is displayed.
2. Click **Create** to create a virtual port.
   The Create Virtual Port page is displayed.
3. Enter a name for the virtual port in the Port Name field.
4. Select a network from the Network list that you want to associate with the virtual port.
5. Select a security group from the Security Group list that you want to apply to the virtual port.
6. Select floating IPs from the Floating IPs list that you want to associate with the virtual port.

7. To add an SLO, click the Advanced Options section and select an SLO from the Security Logging Object(s) list.
8. Click **Create** to create the virtual port.

Attaching an SLO to an existing Virtual Machine Interface

Follow these steps to attach an SLO to an existing virtual machine.

1. Navigate to **Overlay**>**Virtual Ports**.

   The Virtual Ports page is displayed.
2. Select the virtual port by selecting the check box next to the name of the virtual port, and click the **Edit** icon.

   The Edit Virtual Port page is displayed.
3. To add an SLO, click the Advanced Options section and select an SLO from the Security Logging Object(s) list.
4. Click **Save** to save configuration.

Editing an Existing SLO

Follow these steps to edit the name of an existing SLO.

1. Navigate to **Security**>**Security Logging Object**.

   The Security Logging Object page is displayed.
2. To edit an existing SLO, click the **Edit** icon at the end of the row.
3. Update the necessary information.
4. Click **Save** to save configuration.

System Log Receiver in Contrail Analytics

IN THIS SECTION

- Overview | **105**
- Redirecting System Logs to Contrail Collector | **105**
- Exporting Logs from Contrail Analytics | **105**

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 `common_vrouter.env` can be configured to send flow logs to external syslog server. You can configure `common_vrouter.env`, if you wish to recompose the docker. Update `common.sh` , if you do not wish to recompose the docker.

If you wish to recompose the docker:

1. Update `/etc/contrail/common_vrouter.env`.

   This is applicable only for the docker-based deployment.

```
SLO_DESTINATION=syslog
SAMPLE_DESTINATION=syslog
```

2. Recompose the docker.

```
docker-compose -f /etc/contrail/vrouter/docker-compose.yaml down

docker-compose -f /etc/contrail/vrouter/docker-compose.yaml up -d
```

3. Configure the session export rate.

If you do not wish to recompose the docker:

1. Update `common.sh` .

```
SLO_DESTINATION="syslog file"
SAMPLE_DESTINATION="syslog file”
if [ -n "$XFLOW_NODE_IP" ];
```

2. Configure the session export rate.

3. Then restart contrail-vrouter-agent.

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 | **107**
- Examples: Detecting Anomalies | **109**
- Configuring the User-Defined Log Statistic | **111**
- Implementing the User-Defined Log Statistic | **114**

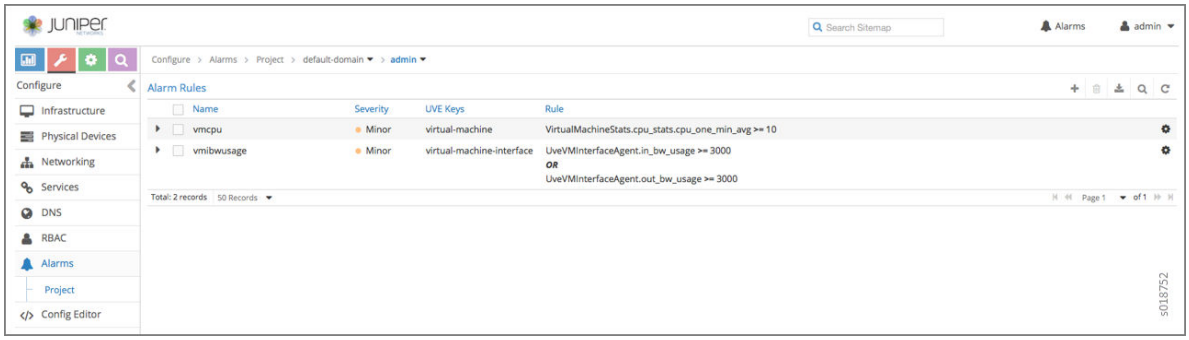
Configuring Alarms Based on User-Visible Entities Data

Contrail allows you to 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 6 on page 108 for field descriptions.

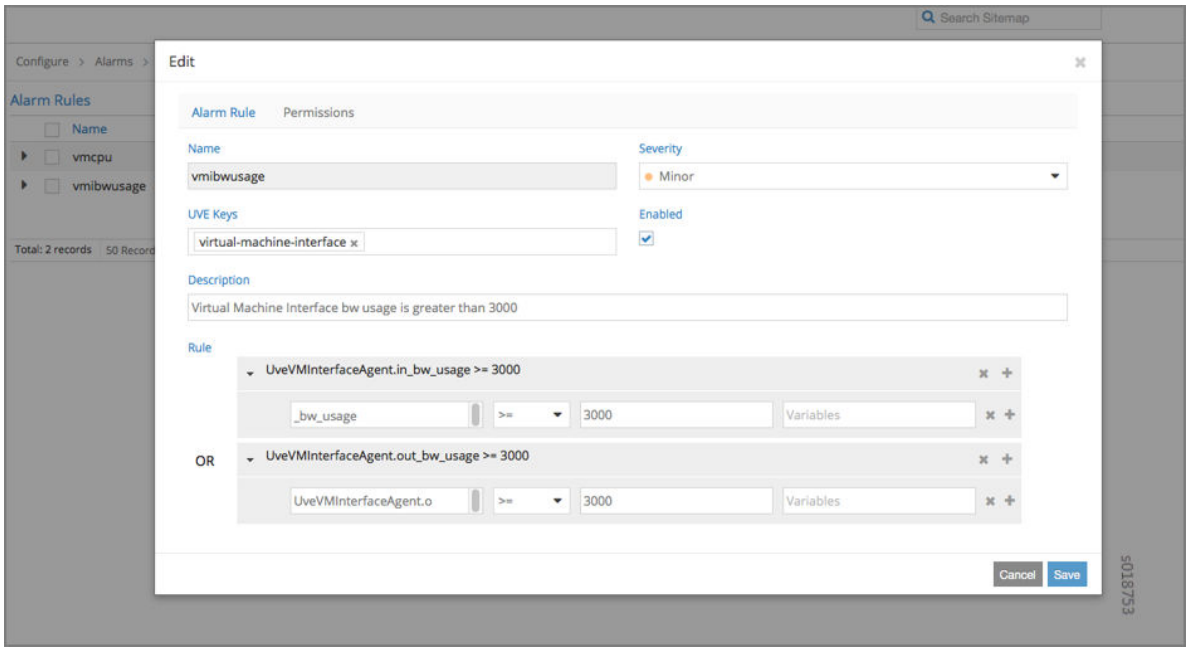


Table 6: Alarm Rules Fields

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.

Table 6: Alarm Rules Fields *(Continued)*

Field	Description
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.

3. 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.

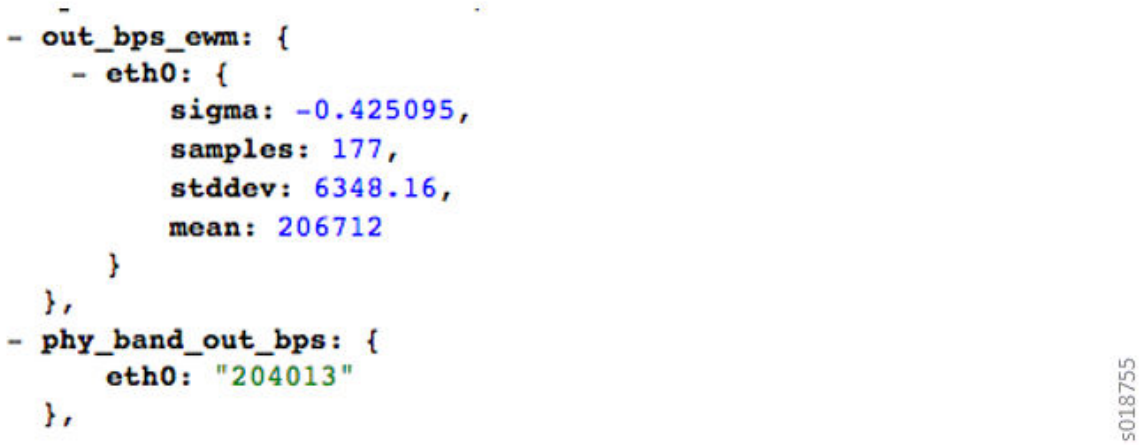
Contrail uses a statistical process control model for time-series anomaly detection that can be computed online, in real-time. Raw metrics are sent as statistics by Sandesh generators embedded inside the UVEs. The model uses the running average and running standard deviation for a given raw metric. The 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`.



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.

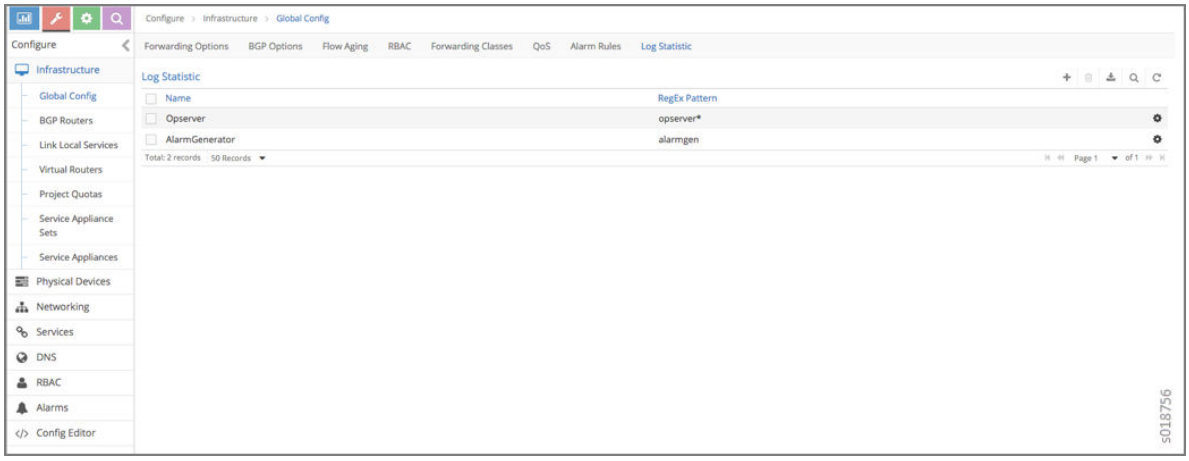
Contrail analytics stores all of the monitored messages in the Contrail database node, and the analytics generates a large amount of useful information that aids in monitoring and troubleshooting the network.

With Contrail, 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.

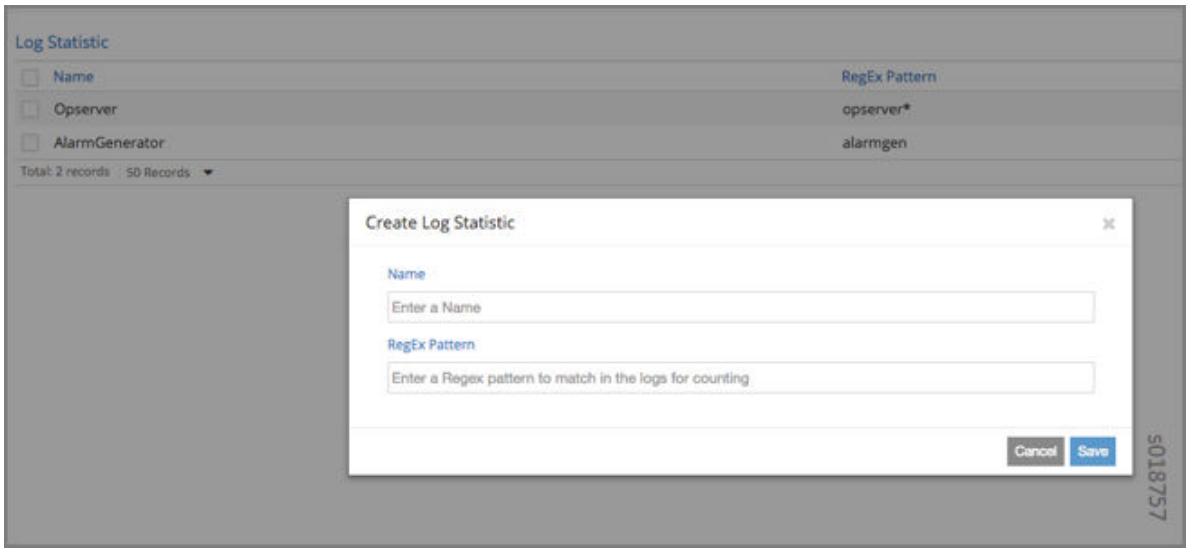
The user-defined log statistic can be configured from the Contrail UI or from the command line, using `vnc_api`.

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

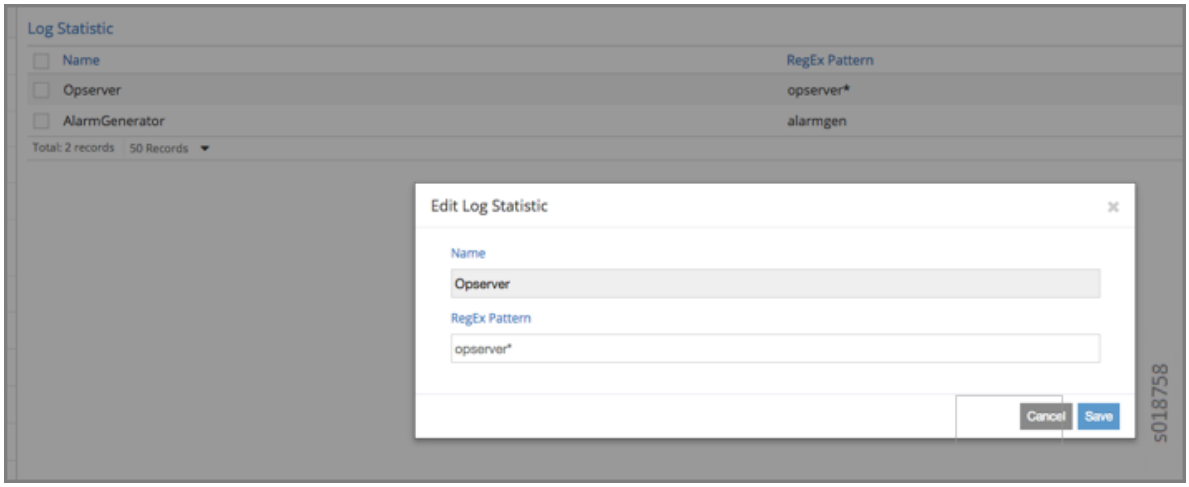
1. Navigate to **Configure > Infrastructure > Global Config** and select **Log Statistic**.



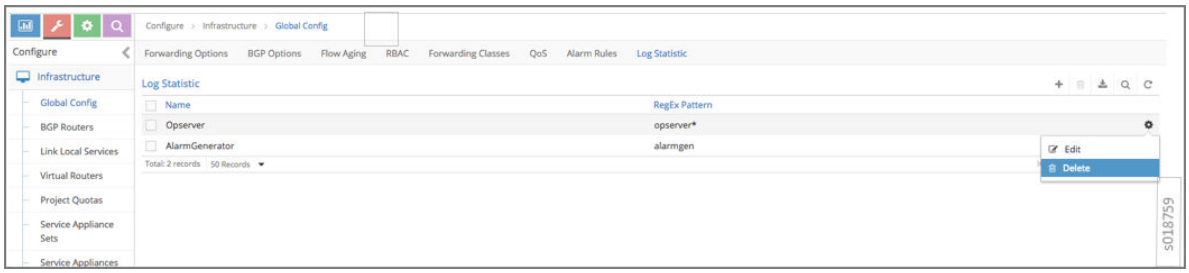
2. To create a log statistic, click the plus (+) icon to access the **Create Log Statistic** screen. Enter a name for the user-defined log statistic, and in the **RegExp Pattern** field, enter the Perl-type expression to look for and count.



3. 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.



4. 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
}
    ]
}
```

Contrail Networking Alarms

Table 7 on page 117 lists the default alarms in Contrail Networking and their severity levels.

An alarm with severity level 0 (zero) is critical, 1 (one) is major, and 2 (two) is minor.

Table 7: Contrail Networking Alarms and Severity Level

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-address-mismatch-compute	1	Compute Node IP Address mismatch.	The compute node IP address provided in the configuration file and the IP address provided as part of creating (provisioning) vrouter-agent do not match.
system-defined-address-mismatch-control	1	Control Node IP Address mismatch.	IP address for control node is different in config node and control node.
system-defined-bgp-connectivity	1	BGP peer mismatch. Not enough BGP peers are up.	Total number of BGP peers is different from the configured number of BGP peers.

Table 7: Contrail Networking Alarms and Severity Level *(Continued)*

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-bottle-request-size-limit	-	Bottle request size limit exceeded.	Request Size received by API server is too large. In most cases, this can be resolved by increasing the value set for the variable *max_request_size* in the **/etc/contrail/contrail-api.conf** file in config API Docker container. However, as a good practice, investigate as to why such a huge request is being sent to the Config API server.
system-defined-conf-incorrect	1	ContrailConfig missing or incorrect. Configuration pushed to Ifmap as ContrailConfig is missing or incorrect.	Config node did not send ContrailConfig for this node. This could be due to name mismatch between the node configured compared to actual node.
system-defined-disk-usage-high	1	Disk usage exceeds high threshold limit.	Corresponding disk is filled between 70%-90% capacity. Delete some files to create disk space.
system-defined-disk-usage-critical	0	Disk usage crosses critical threshold limit.	Corresponding disk is filled up > 90%. Delete some files to create disk space.
system-defined-node-status	0	Node Failure. NodeStatus UVE not present.	NodeStatus UVE is not present or process is non-functional for this node. Verify that the `process` and `nodemgr` is up.
system-defined-partial-sysinfo	1	System Info Incomplete.	build_info is not present in NodeStatus. Cause unknown at this time.

Table 7: Contrail Networking Alarms and Severity Level *(Continued)*

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-process-connectivity	0	Process(es) reporting as non-functional.	One or more processes have connections missing.
system-defined-process-status	0	Process Failure.	Review the docker logs to understand the reason for process failure.
system-defined-prouter-connectivity	1	Prouter connectivity to controlling tor agent does not exist. Contrail looks for non-empty value for connected_agent_list	Check for OVSDB connectivity status on the physical device. Debug for link failures between physical device or OVSDB connection failure between the vrouter-agent and physical router.
system-defined-prouter-tsn-connectivity	1	Prouter connectivity to controlling TSN agent does not exist. Contrail looks for non-empty value for tsn_agent_list.	Check for OVSDB connectivity status on the physical device. Debug for link failures between physical device or OVSDB connection failure between the vrouter-agent and physical router.
system-defined-storage-cluster-state	1	Storage Cluster warning or errors.	Since Contrail is not provisioning storage this alarm is not generated.

Table 7: Contrail Networking Alarms and Severity Level *(Continued)*

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-vrouter-interface	1	vRouter interface(s) down.	This alarm is raised if forwarding and bridging is disabled or if health check has failed. Other reasons for this alarm include the following: • no IP or subnet assignment • admin state is down • parent interface is down • VLAN is down • oper state is down • config is missing Resolve above items based on information available from the introspect page for interface.
system-defined-xmpp-connectivity	1	XMPP peer mismatch.	Number of XMPP peers is different from configured XMPP peers.
system-defined-xmpp-close-reason	1	XMPP connection closed towards peer. Alarm has reason to close.	This alarm is deprecated.
system-defined-core-files	0	A core file has been generated on the node.	There is some core file in the node.
system-defined-pending-cassandra-compaction-tasks	1	Pending compaction tasks in cassandra crossed the configured threshold.	This alarm is raised when disk space is insufficient. Check Cassandra system logs to understand the reason for pending compaction.

Table 7: Contrail Networking Alarms and Severity Level *(Continued)*

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-package-version-mismatch	0	There is a mismatch between installed and running package version.	Package version for the package mentioned in the alarm is not matching with the required version.
system-defined-vrouter-limit-exceeded	1	Agent resource usage exceeded configured watermark for resource.	This alarm is raised when the next hop count or the used mpls label count crosses the high watermark. The alarm is reset when the next hop count or the used MPLS label count becomes less than the low watermark. To reset alarm, delete the `nexthop` and `mpls` label, which can be achieved by deleting virtual machines on the compute. Alarm can also be cleared by increasing the default watermark, which is 80 (80% of the maximum number of nexthops configured in vRouter after which alarm is raised). For this, you need to change the configuration in the **contrail-vrouter-agent.conf** file and restart the vRouter agent.

Table 7: Contrail Networking Alarms and Severity Level *(Continued)*

Alarm Name	Severity	Description	Steps to Resolve This Alarm
system-defined-vrouter-table-limit-exceeded	0	Agent resource usage exceeded table size for resource in vRouter.	This alarm is raised when the next hop count reaches the `nexthop count` configured in vRouter, or when the maximum number of MPLS labels on the compute are used. This alarm is cleared when the next hop count goes below 95% of the next hop count in vRouter, or the number of used MPLS label count becomes 95 % of the maximum labels or less. To reset the alarm, delete the nexthop and MPLS labels, which can be achieved by deleting virtual machines on the compute for which alarm is raised. This alarm can also be reset by increasing the maximum number of `nexthop` and MPLS labels configured in vRouter, if it is not already configured to the maximum supported limit.

Alarms History

IN THIS SECTION

- Viewing Alarms History | **123**

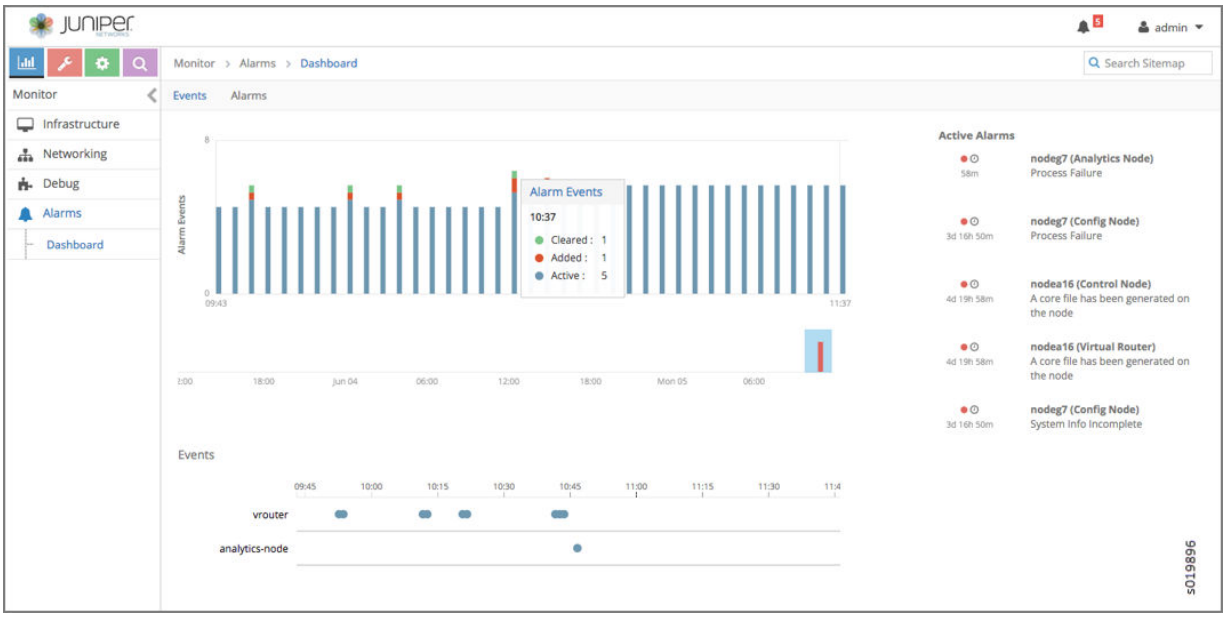
Contrail allows you to view a history of alarms that were raised or reset. You can also view a history of user-visible entities (UVEs) that have been changed.

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 15 on page 123 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.

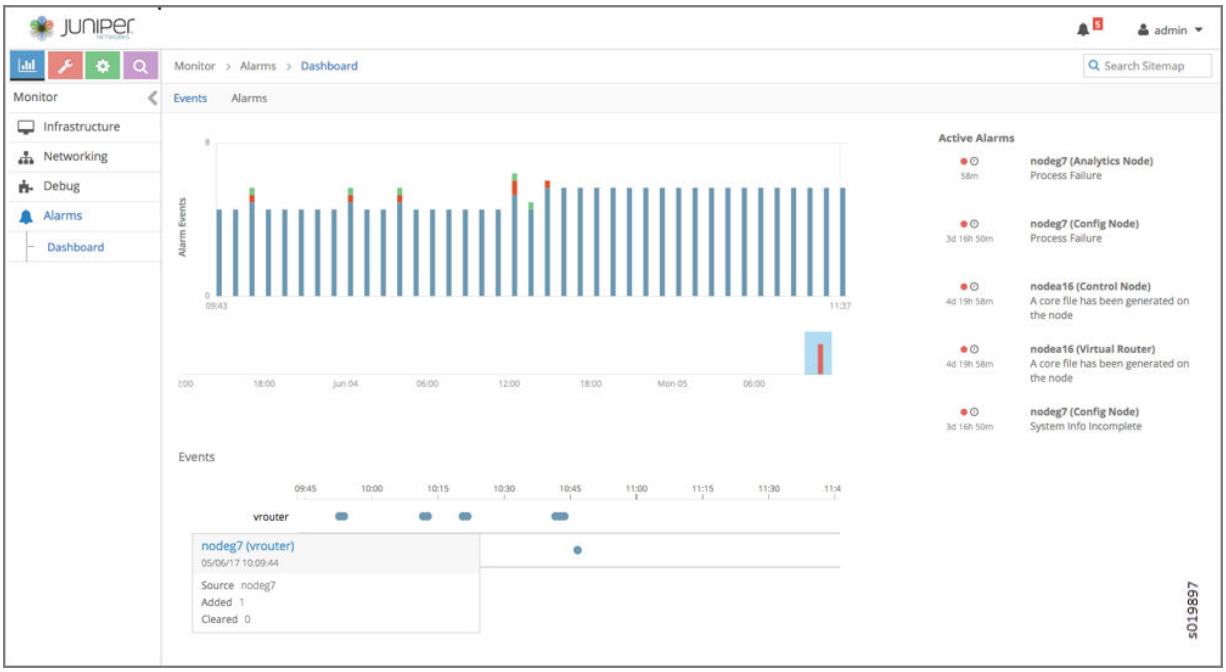
You can also use a `contrail-status` query to view the alarms history. Additionally, the `contrail-status` displays a history of added, updated, and removed information for UVEs in Contrail.

Figure 15: Alarms History Page



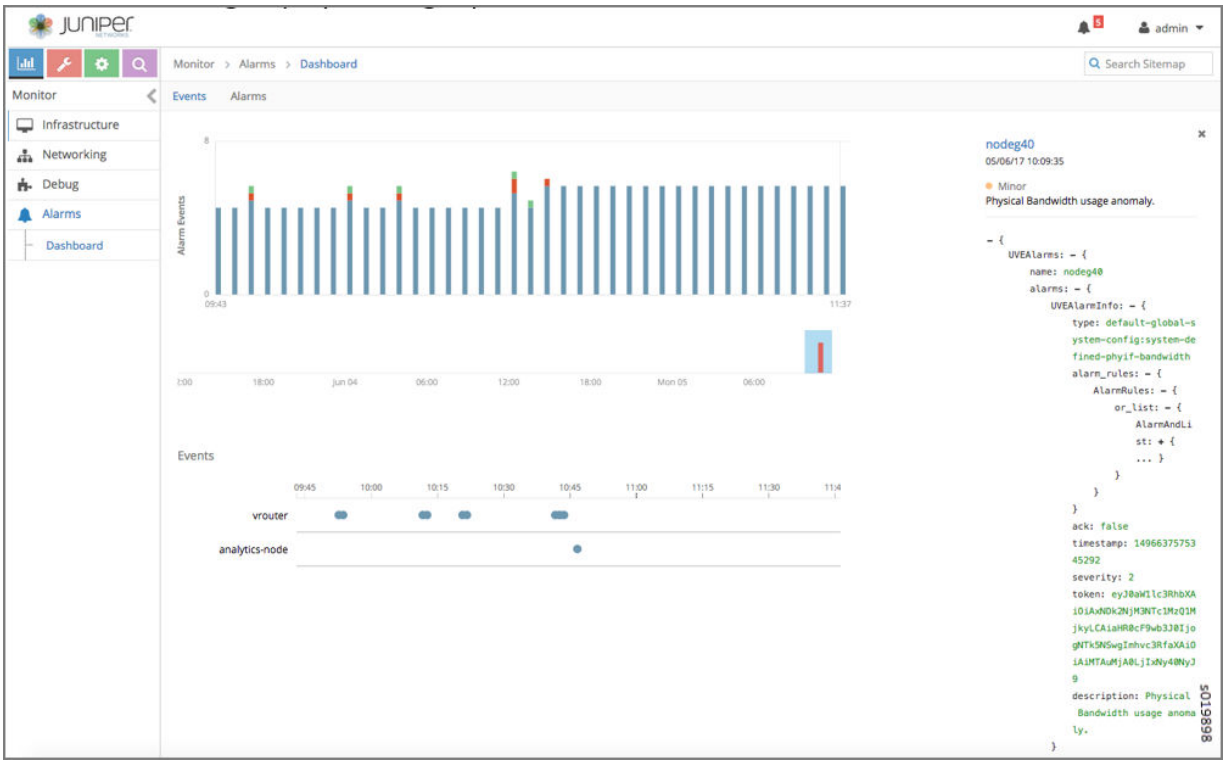
Tooltips are available on the Alarms History page. In the Events area, you can click on any node type listed to display a tooltip showing details of the events that have been added and cleared in that node, see Figure 16 on page 124 .

Figure 16: 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 17 on page 125 .

Figure 17: Detailed Event Log



RELATED DOCUMENTATION

User Configuration for Analytics Alarms and Log Statistics | 107

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 | 107

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.

With Contrail, 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 | 107

Configuring Analytics as a Standalone Solution

IN THIS SECTION

Overview: Contrail Analytics as a Standalone Solution | 127
Configuration Examples for Standalone | 127

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

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

Examples: Inventory File Controller Components

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

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 | 142

Understanding Contrail Analytics | 2

Agent Modules in Contrail Networking

IN THIS SECTION

- Config | 132
- Oper-DB | 133
- Controller | 139
- Agent KSync | 141
- UVE | 141
- Services | 141

The VNsw Agent (also called Agent) in Contrail Networking is responsible for managing the data plane component. It is similar to any datapath agent that runs on the line cards of a network node. Agent responsibilities include:

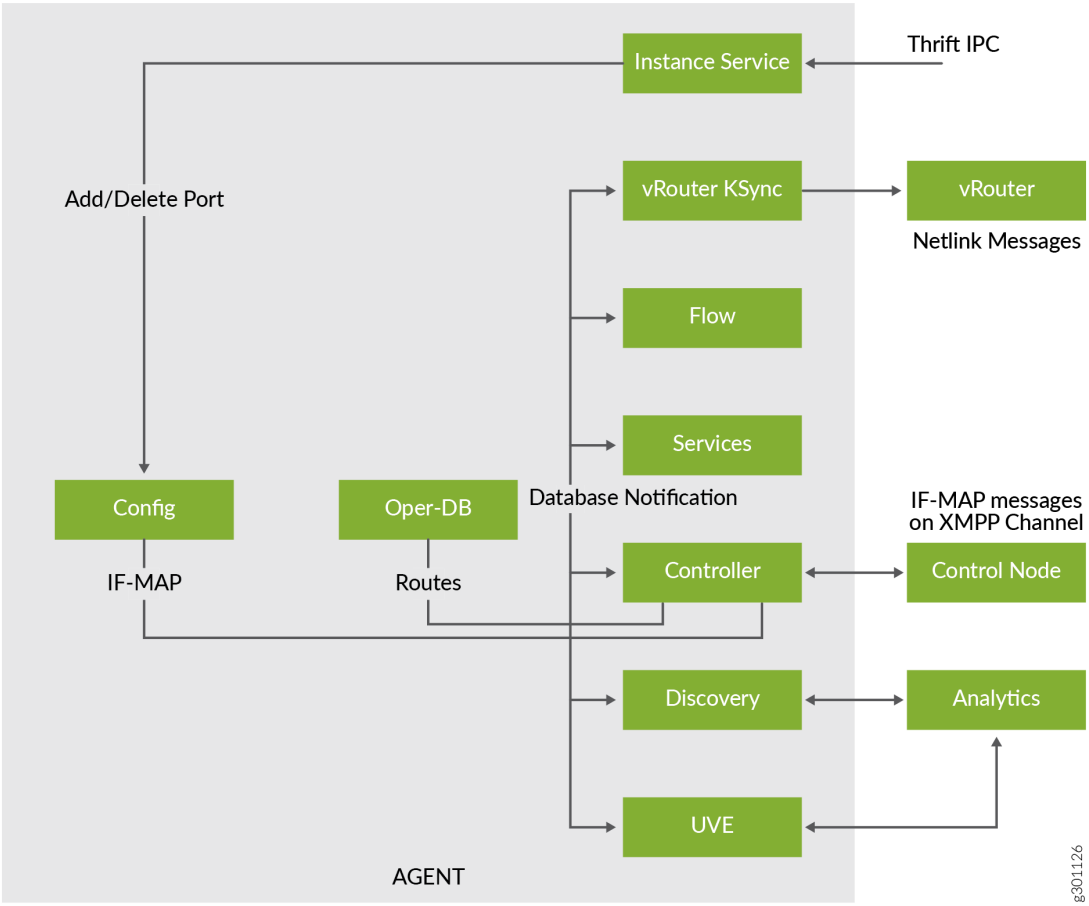
- Interface with `contrail-controller` to get the configuration. Agent receives the configuration and translates it into a form that the datapath can understand.
- Interface with `contrail-controller` to manage routes.
- Collect and export statistics from datapath.
- Translate the data model from IF-MAP to the data model used by datapath.

Agent contains the following modules:

- Config
- Oper-DB
- Controller
- UVE
- Pkt
- Services
- KSync

Agent by itself is not a program or daemon. Based on the platform, daemons are built using the modules listed above. The `contrail-vxlan-agent` is the port of `contrail-vrouter-agent` on platforms supporting VXLAN bridges. Figure 18 on page 132 provides an overview of the different modules involved.

Figure 18: Overview of Agent Modules



Config

Config module implements the northbound interface for Agent. Agent gets two types of configurations, virtual machine ports and IF-MAP.

Virtual-Machine Ports

Agent opens a thrift service (name InstanceService) to listen for Port-Add/Port-Delete message. Port-Add informs agent about a virtual machine (VM) interface created on the compute node. The Port-Add message also contains the following information:

- Name of virtual machine port.
- Virtual machine for the port.
- Mac and IP address for the port.

- Virtual network for the port.

Once agent knows about the creation of a port, it will send a subscribe message to `contrail-controller` for `virtual-machine`. When `contrail-controller` receives the subscribe message for a `virtual-machine`, it walks through the IF-MAP graph and sends all configuration relevant for the `virtual-machine` to the Agent. The module invoking port add message is platform dependent. In the case of OpenStack, `nova-compute` service invokes the message.

IF-MAP

All of the Contrail Virtual Network Controller (VNC) configuration is stored as a Metadata Access Point (MAP) database. The MAP database is accessed using IF-MAP protocol.

Agent does not access the MAP database directly. Instead, Agent opens an XMPP connection to `contrail-controller` to get the MAP configuration. The `contrail-controller` works on a subscription model. Agent must subscribe to the virtual machines of interest and `contrail-controller` will download all of the configuration relevant to the `virtual-machine`. As a result, Agent receives only the minimal configuration needed. Agent subscribes to a `virtual-machine` when it receives a port add message for a `virtual-machine-interface`.

Agent uses the `ifmap-agent-client` library to parse the IF-MAP messages from the XMPP channel to the `contrail-controller`. The `ifmap-agent-client` defines a `DBTable` for every IF-MAP node type. A special `DBTable` is defined to store the IF-MAP links. The `ifmap-agent-client` also creates a graph for ease of navigating the IF-MAP configuration. An IF-MAP node is vertex in the graph and links form the edges in the graph.

Configuration Management

Config module registers `DBTables` of interest from the `ifmap-agent-client` library. Any add, delete, or update of the configuration results in a callback to the Config module. The Config module then does basic validation on the config nodes and then triggers the operational module to process the configuration.

Redundancy

Agent connects to two different control nodes for redundancy. When the XMPP connection for one of the control node fails, it will subscribe to the other control node for configuration. When connecting to the new control node, Config module audits the configuration to remove stale configuration.

Oper-DB

The Oper-DB module holds the operational state of the different objects in Agent. The operational state processes the configuration and creates different tables appropriate for Agent.

Following are the principal tables in Oper-DB:

Virtual Network

Table of all `virtual-networks` with UUID as the key. It contains the following information:

Table 8: Virtual Network Table

Item	Description
VRF	The `routing-instance` for the `virtual-network`.
IPAM Data	The IP Address Management (IPAM) configured for the `virtual-network`. It includes DHCP configuration, DNS configuration, subnet configuration, and so on.
Network Policy	Network policy access control list (ACL) for the `virtual-network`.
Mirroring	Mirroring ACL for the `virtual-network`.
VXLAN-ID	Virtual Extensible Local Area Network ID (VXLAN-ID) to be used when VXLAN encapsulation is used.
Layer 3 Forwarding	Specifies if `layer3_forwarding` is enabled for IPv4 and IPv6 packets.
Bridging	Specifies if bridge forwarding is enabled. Even if `layer3_forwarding` is disabled, IPv4 and IPv6 packets are bridge forwarded.

VRF

The virtual routing and forwarding (VRF) table represents a `routing-instance` in configuration. Each `virtual-network` has a "native" VRF. Other than the per `virtual-network` VRF, there can be other internal VRFs. The internal VRFs are used in features, such as service chaining.

Each VRF has a set of routing tables as its members.

Table 9: VRF Routing Tables

Table	Description
Inet4 Unicast Table	Table containing inet4 unicast routes.
Inet4 Multicast Table	Table containing inet4 multicast routes.
EVPN Table	Table containing EVPN routes keyed with MAC address, IP address, and `vxlan/ethernet_tag`.
Bridge Table	Table containing MAC addresses. The bridge table is currently used only in the case of a "native" VRF for a `virtual-network`.
Based the platform used, Agent creates some VRFs implicitly:	
OpenStack	Agent implicitly creates a VRF for `fabric-network` with the name `default-domain:default-project:ip-fabric:__default__`.
Xen	Agent implicitly creates a VRF for `fabric-network` with the name `default-domain:default-project:__link_local__`.

Virtual Machine

The virtual machine table stores all `virtual-machines` created on the compute node.

Interface

The interface table contains all of the interfaces in Agent. Based on the type of interface, the trigger to create an interface can vary. Also, the key fields used to uniquely identify the interface and the data fields in an interface can vary based on the type of interface.

Agent supports the following different types of interfaces:

Table 10: Interface Types Supported by Agent

Item	Description
Physical Interface	Represents physical ports on the compute node. Physical interfaces are created based on the `config-file` for Agent. Key for physical interface is *<interface-name>*.
Packet Interface	Interface used to exchange packets between vRouter and Agent. Typically named `pkt0`, this interface is automatically created in Agent. Key for packet interface is *<interface-name>*.
Inet interface	he layer 3 inet interfaces are managed by Agent. Agent can have one or more inet interfaces based on the platform used. • OpenStack: In the case of OpenStack, Agent creates the `vhost0` inet-interface. `vhost0` is a layer 3 interface in `host-os`. Agent uses this layer 3 interface for tunnel encapsulation and decapsulation. The interface is added into the fabric VRF. • Xen: In the case of Xen, Agent creates the `xapi0` interface. The `xapi0` interface is added into the Xen `link-local` VRF. • vGW: Every vGW Virtual Gateway instance has a vGW interface created. The vGW interface is an unnumbered interface and does not have an IP address. Key for inet interface is *<interface-name>*.
VM Interface	This interface represents a `virtual-machine-interface`. The interface is created when Agent receives an `AddPort` message from the Apache Thrift service `InstanceService`. Key for VM interface is UUID for the interface.

An interface is in **Active** state if all of the necessary configuration for the interface is available and it can be made operational.

An interface is in **Inactive** state if it cannot be made operational. The reason can be missing configuration, the `link-state` down, and so on.

Routes

Every VRF has a set of routing tables for inet4 unicast routes, inet4 mulitcast routes, EVPN routes, and bridge MAC entries.

Every route specifies the forwarding action for a destination. Agent has multiple modules that can have different views of forwarding action for a destination. Each forwarding action is specified in the form of a path. Each module that adds a path is identified by a peer in the path.

Route keeps the list of paths sorted. The head of this list is treated as the **Active** path for the route.

Every path contains next hop that describes forwarding action.

The unicast routing table also maintains route entries in the Patricia tree form to support longest prefix match (LPM) on the tree.

Next Hop

Next hop describes the forwarding action for routes pointing to it. When route lookup for an address hits the route, the forwarding action for the packet is defined by the next hop.

The different types of next hop supported in Agent are:

Table 11: Next Hop Types Supported by Agent

Type	Description
Discard	Packets hitting `Discard` next hop must be dropped.
Receive	Packets hitting `Receive` next hop are destined to the `host-os`. The next hop has an interface on which packets must be transmited.
Resolve	Packets hitting `Resolve` next hop need ARP resolution. For example, if IP address 10.1.1.1/24 is assigned to interface `vhost0`, the following routes and next hop are generated. • Route 10.1.1.1/32 is added with `Receive` next hop pointing to `vhost0`. • Route 10.1.1.0/24 is added with `Resolve` next hop. Any packet hitting this route triggers ARP resolution.
ARP	Routes created as a result of ARP resolution, that point to ARP next hop. In the example above, you can have routes 10.1.1.1.2/32, 10.1.1.3/32, and so on pointing to ARP next hop.

Table 11: Next Hop Types Supported by Agent *(Continued)*

Type	Description
Interface	Specifies that packets hitting this next hop must be transmitted on the interface.
Tunnel	Specifies that packets hitting this next hop must be encapsulated in a tunnel. The tunnel next hop specifies tunnel destination IP address. The packet post tunneling is routed on the fabric network.
Multicast Composite	Mulitcast composite next hop contains a list of component next hops. Packets hitting the multicast composite next hop are replicated and transmitted on all the component next hops.
ECMP Composite	Equal Cost Multi-Path (ECMP) composite next hop contains a list of component next hops. Packets hitting the ECMP composite next hop must be sent out on one of the component next hops. Packet forwarding component must ensure that packets for a connection are always transmitted on the same component next hop of a ECMP composite next hop. ECMP composite next hop is used to load balance traffic across multiple next hops.

MPLS

The MPLS label defines the forwarding action for MPLS tunneled packets received on the fabric network.

Agent assigns the following labels:

- Two labels are allocated for every VM interface.
  - A label for layer 3 packets.
  - A label for bridge packets.
- A label for every ECMP composite next hop.
- A label for every multicast composite next hop.

The `label-range` for multicast composite next hop is preallocated and does not overlap with other labels.

Multicast

Multicast module is responsible for managing multicast routes.

VXLAN

The VXLAN table contains an entry for every VXLAN ID created.

Controller

This module manages the communication between Agent and `contrail-controller`. Agent connects to two Contrail controllers for redundancy. Two XMPP channels are opened with each of the Contrail controllers.

Configuration Channel

The Contrail controller uses this channel to send IF-MAP configuration to Agent. Agent subscribes to configuration from only one of the XMPP channels at a time. If the subscribed channel fails, it will switch subscription to the other channel.

Route Channel

This channel is used to exchange routes between Agent and Contrail controller. Agent connects to two Contrail controllers at a time and routes are exchanged between both of the channels. Routes from each of the channels is added with a different "Route Peer." When one of the channels fails, it only deletes "Route Path" from the channel that failed.

Route Export

Agent exports routes for `virtual-machines` created on the local compute node. Agent exports the route with the following information:

- Routing instance for the route.
- Destination network for the route (also called a `route-prefix`).
- Next hop information:
  - MPLS label for route if MPLSoGRE or MPLSoUDP encapsulation is used.
  - VXLAN ID for route if VXLAN encapsulation is used.
  - Gateway for the route. This is implicitly derived from the XMPP channel.
  - Security group membership for the routes.

The control node implicitly derives the `virtual-network` name for the route from the `routing-instance`.

Route Import

Agent subscribes to all `routing-instances` in the VRF table. The `contrail-controller` collects routes from all Agents. Controller synchronizes routes in a `routing-instance` if Agent is subscribed to the `routing-instance`.

Routes are exchanged between Agent and `contrail-controller` over the XMPP channel in XML format.

Controller module decodes the XMPP messages and adds or deletes "Route Paths" into the routing tables. The `contrail-controller` provides the following information for every route:

- Routing instance for the route.
- Destination network for the route.
- MPLS label for the route if MPLSoGRE or MPLSoUDP encapsulation is being used.
- VXLAN ID for route if VXLAN encapsulation is used.
- Gateway for the route. This is implicitly derived from the XMPP channel.
- Security group membership for the routes.
- Virtual network for the route.

The `contrail-controller` also reflects back the routes added by Agent itself. When the route is received, Agent looks at the gateway IP address to identify if the route is hosted on a local compute node or a remote compute node. If the route is hosted on a remote compute node, the Controller module creates a next hop tunnel to be used in route. If the route is hosted on a local compute node, a route pointing to the next hop interface is added.

Headless Mode

When the XMPP channel from Agent to the Contrail controller fails, Agent flushes all of the "Route Paths" added by the controller. If the connection to both of the Contrail controllers fail, this can result in deleting routes distributed by the controller.

Connections to Contrail controllers can fail for many reasons including network failure, Contrail controller node failing, and so on. Deleting paths can result in connectivity loss between virtual machines.

Headless mode is introduced as a resilient mode of operation for Agent. When running in headless mode, Agent retains the last "Route Path" from Contrail controller. The "Route Paths" are held until a new stable connection is established to one of the Contrail controllers. Once the XMPP connection is up and is stable for a predefined duration, the "Route Paths" from the old XMPP connection are flushed.

Agent KSync

Oper-DB in Agent contains different tables and defines the data model used in the Agent. While the Agent data model was initially developed for Contrail vRouter agent, it is mostly independent of the underlying forwarding platform.

The data model used by datapath can vary based on the platform being ports. Agent KSync module is responsible to do the translation between the data model used by Agent and the datapath.

The functionality of Agent KSync includes:

- Provide translation between the data model of Agent and the forwarding plane.
  - KSync will be aware of the data model used in the data plane.
  - Oper-DB defines the data module for Agent.
- Keeps the operational state of Agent in sync with the forwarding plane.
- Keep Agent platform independent.

  Ex: KSync in Contrail vRouter agent is the only module that knows which flow table is memory mapped into the Contrail vRouter Agent memory.

UVE

UVE module is responsible for generating UVE messages to the collector. UVE module registers with Oper-DB and also polls the flows/vrouter to generate the UVE messages to the collector.

Services

This module is responsible to run the following services in Agent:

- ARP
- DHCP
- DNS
- Ping
- ICMP error generation

RELATED DOCUMENTATION

vRouter Command Line Utilities | 34

Configuring Secure Sandesh and Introspect for Contrail Analytics

IN THIS SECTION

- Configuring Secure Sandesh Connection | **142**
- Configuring Secure Introspect Connection | **143**

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 12 on page 143 .

Table 12: 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**

RELATED DOCUMENTATION

Agent Modules in Contrail Networking | 130

Debugging Processes Using the Contrail Introspect Feature | 239

CHAPTER 3

Configuring Traffic Mirroring to Monitor

IN THIS CHAPTER

- Configuring Traffic Analyzers and Packet Capture for Mirroring | 144
- Configuring Interface Monitoring and Mirroring | 152
- Mirroring Enhancements | 153
- Analyzer Service Virtual Machine | 155
- Using the Wireshark Plugin to Analyze Packets Between vRouter and vRouter Agent on pkt0 Interface | 158
- Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 163

Configuring Traffic Analyzers and Packet Capture for Mirroring

IN THIS SECTION

- Traffic Analyzer Images | 144
- Configuring Traffic Analyzers | 145
- Setting Up Traffic Mirroring Using Configure > Networking > Services | 145

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 145 .

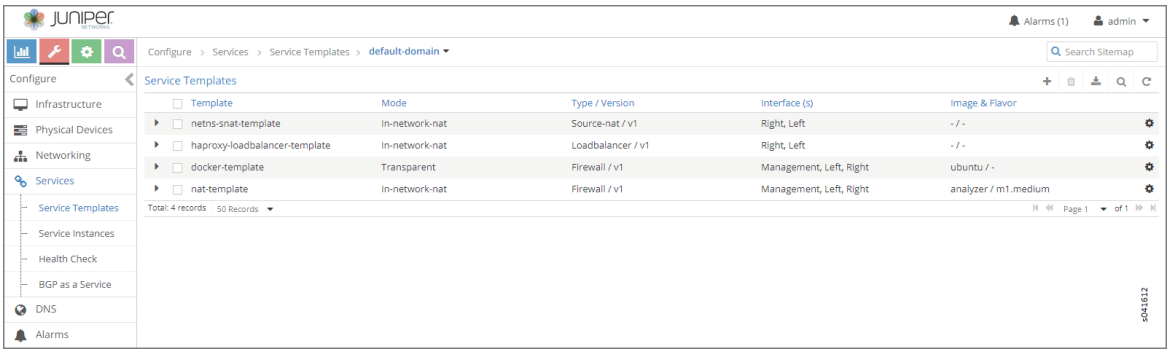
Setting Up Traffic Mirroring Using Configure > Networking > Services

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 19 on page 146 .

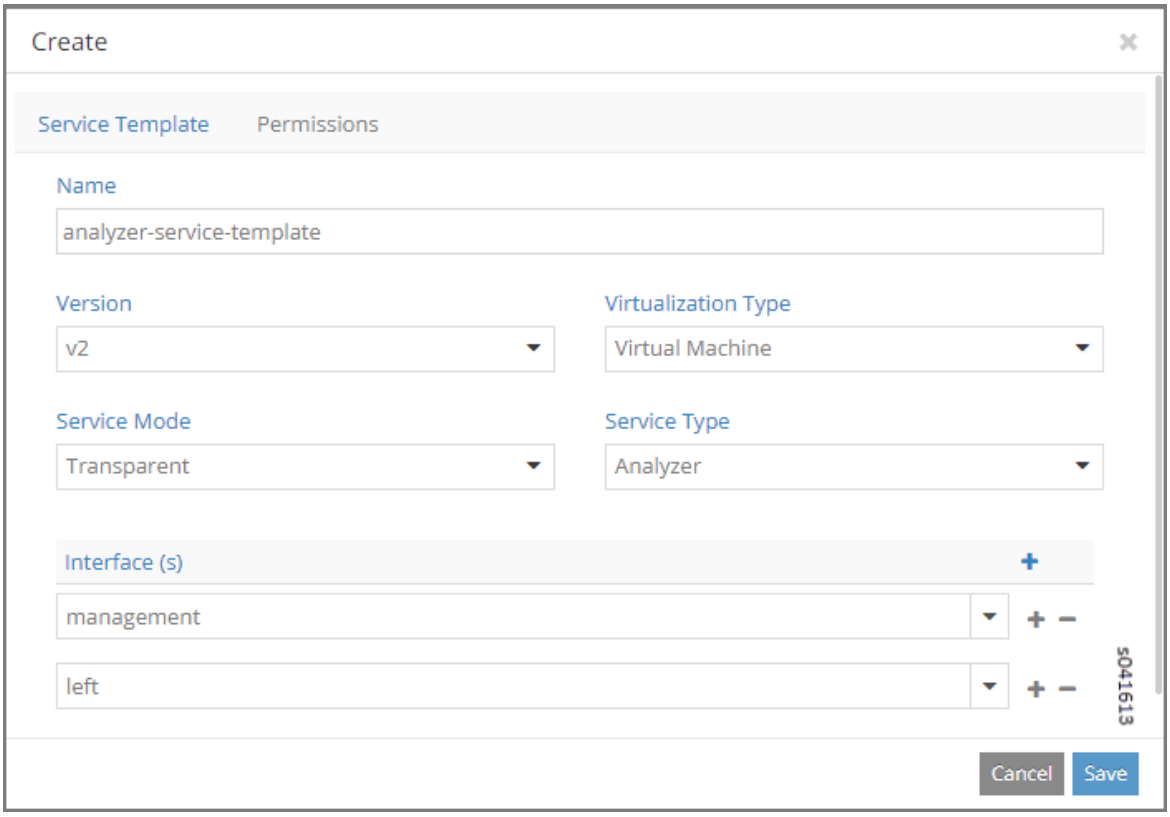
Figure 19: Service Templates



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

   The **Create** window appears. Select the Service Template tab; see Figure 20 on page 146 .

Figure 20: Create Service Template



3. Complete the fields by using the guidelines in Table 13 on page 147 .

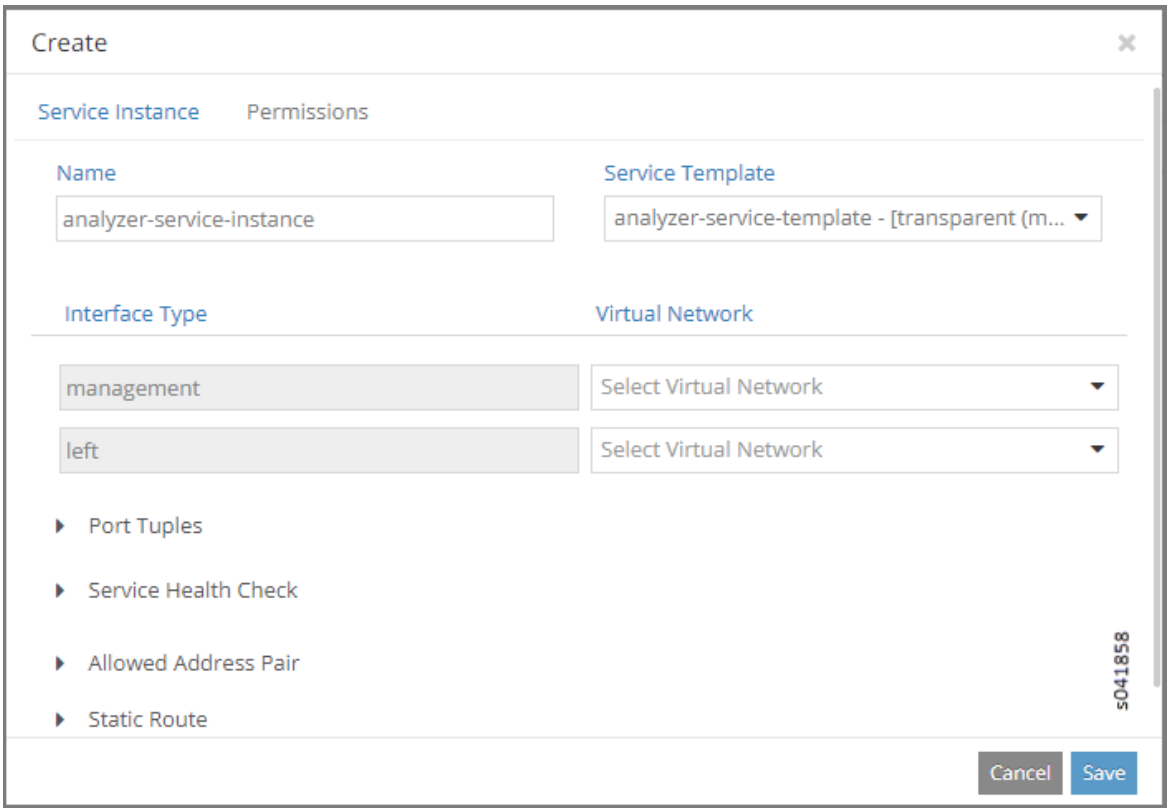
Table 13: Create Service Template Fields

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)	From the drop-down list, click the check boxes to indicate which interface types are used for this analyzer service template: • 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 21 on page 148 .

Figure 21: Create Service Instances



5. Complete the fields by using the guidelines in Table 14 on page 148 .

Table 14: 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.

Table 14: Create Service Instances Fields *(Continued)*

Field	Description
Save	Click **Save** to commit your changes.
Cancel	Click **Cancel** to clear your changes and start over.

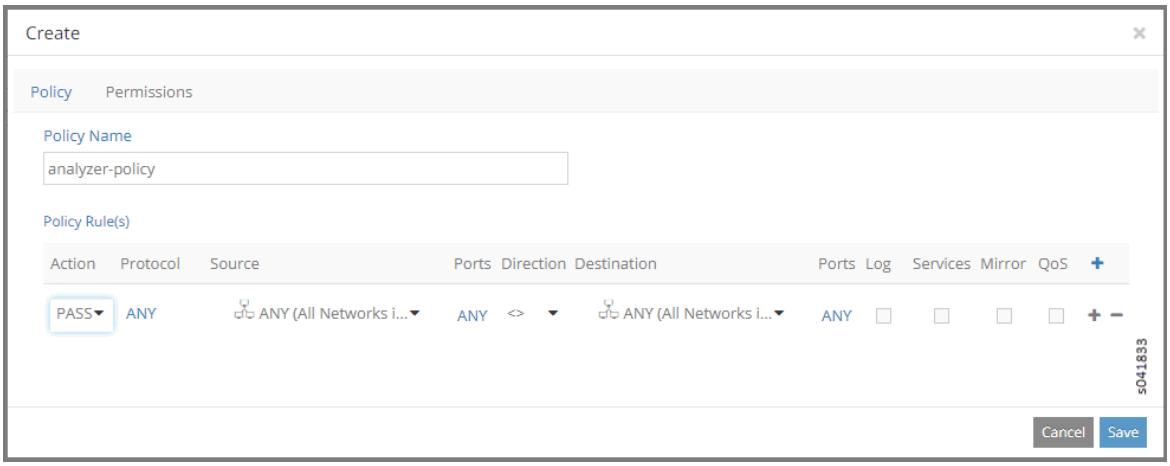
6. To create a network policy rule for this service instance, click **Configure > Networking > Policies**. The **Policies** window appears. Click the **+** icon to get to the **Create** window; see Figure 22 on page 149 .

Figure 22: Create Policy

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 23 on page 150 .

Figure 23: Create Policy Rules



9. To add policy rules, complete the fields, using the guidelines in Table 15 on page 150 .

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

Table 15: 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: • <> (bidirectional) • > (unidirectional)

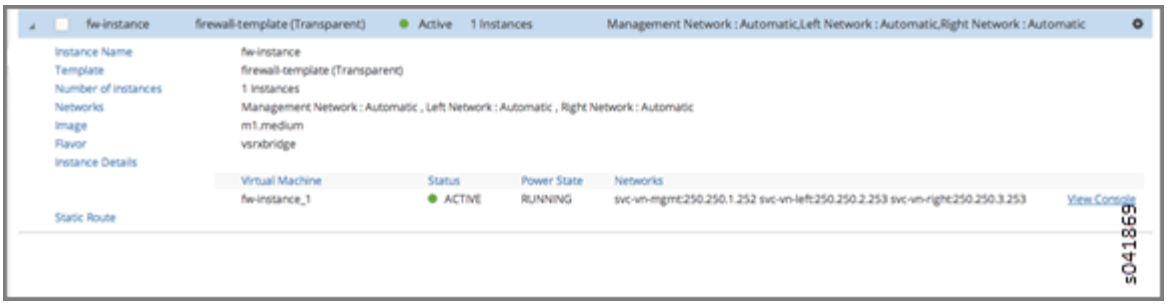
Table 15: Add Rule Fields *(Continued)*

Field	Description
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 24 on page 151 . The analyzer service VM launches the Contrail-enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

Figure 24: Service Instances View Console



RELATED DOCUMENTATION

Configuring Interface Monitoring and Mirroring | 152

Mirroring Enhancements | 153

Analyzer Service Virtual Machine | 155

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 163

Configuring Interface Monitoring and Mirroring

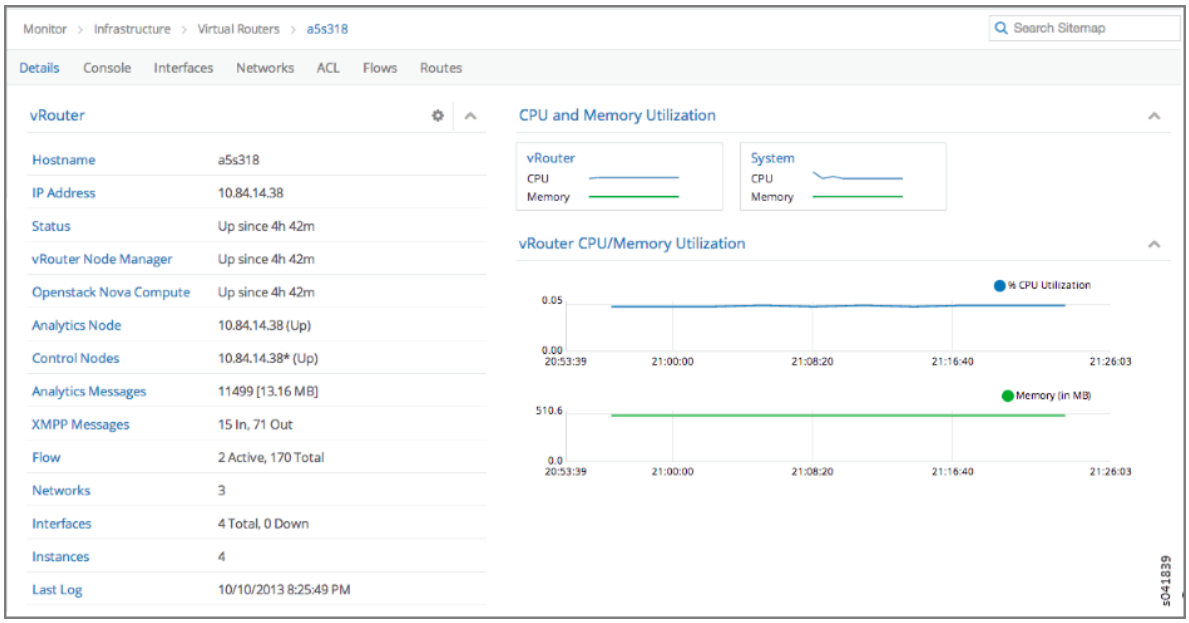
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

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 25 on page 152 .

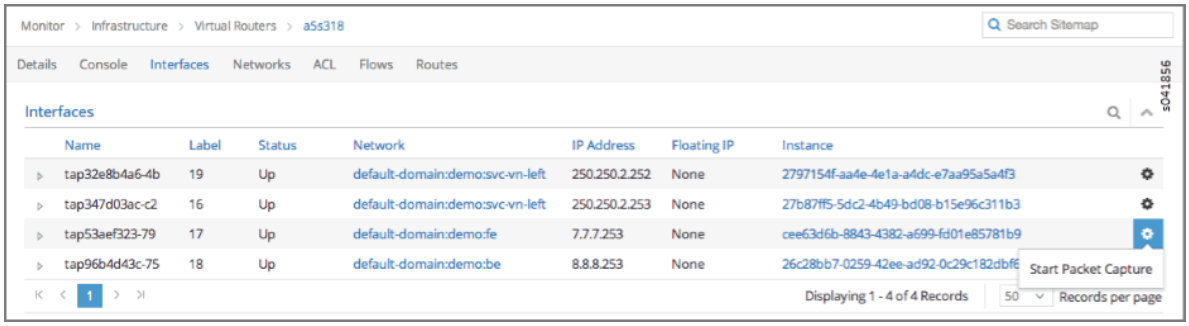
Figure 25: Individual vRouter



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 26 on page 153 .

Figure 26: 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 | 144

Mirroring Enhancements | 153

Analyzer Service Virtual Machine | 155

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 163

Mirroring Enhancements

IN THIS SECTION

- Mirroring Specified Traffic | 154
- Configuring Headers and Next Hops | 154
- How Mirroring is Implemented | 154

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 | 144
- Configuring Interface Monitoring and Mirroring | 152
- Analyzer Service Virtual Machine | 155
- Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 163

Analyzer Service Virtual Machine

IN THIS SECTION

- Packet Format for Analyzer | 155
- Metadata Format | 156
- Wireshark Changes | 157
- Troubleshooting Packet Display | 157

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
- Packet data (original packet data)

The global header is added by the analyzer service by means of the Wireshark instance. The vRouter DP uses the configured UDP session to send mirrored packets to the analyzer, adding the PCAP packet header to the packet data as it sends it over the UDP socket to the analyzer.

The following additional information is also added to the packet data as metadata:

- Captured host (IP address)
- Ingress or egress
- Action (Pass/Deny/...)
- Source VN (fully qualified name)
- Destination VN (fully qualified name)

In the existing PCAP, a network ID is added in the global header. The metadata (additional flow information) is added in front of the existing packet as follows.

+-++-+

| Global header | Packet header| Meta data |Packet data| Packet header| Meta data |Packet data|

+-++-+

Metadata Format

The metadata is in type-length-value (TLV) format as follows.

1. Type: 1 Byte
2. Length: 1 Byte
3. Value: up to length

Type

1. 1 – Captured host IPv4 address
2. 2 - Action field
3. 3 – Source VN
4. 4 – Destination VN
5. 255 – TLV end

Captured host address

Length is 4 or 16 bytes based on IP address type

Action field

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

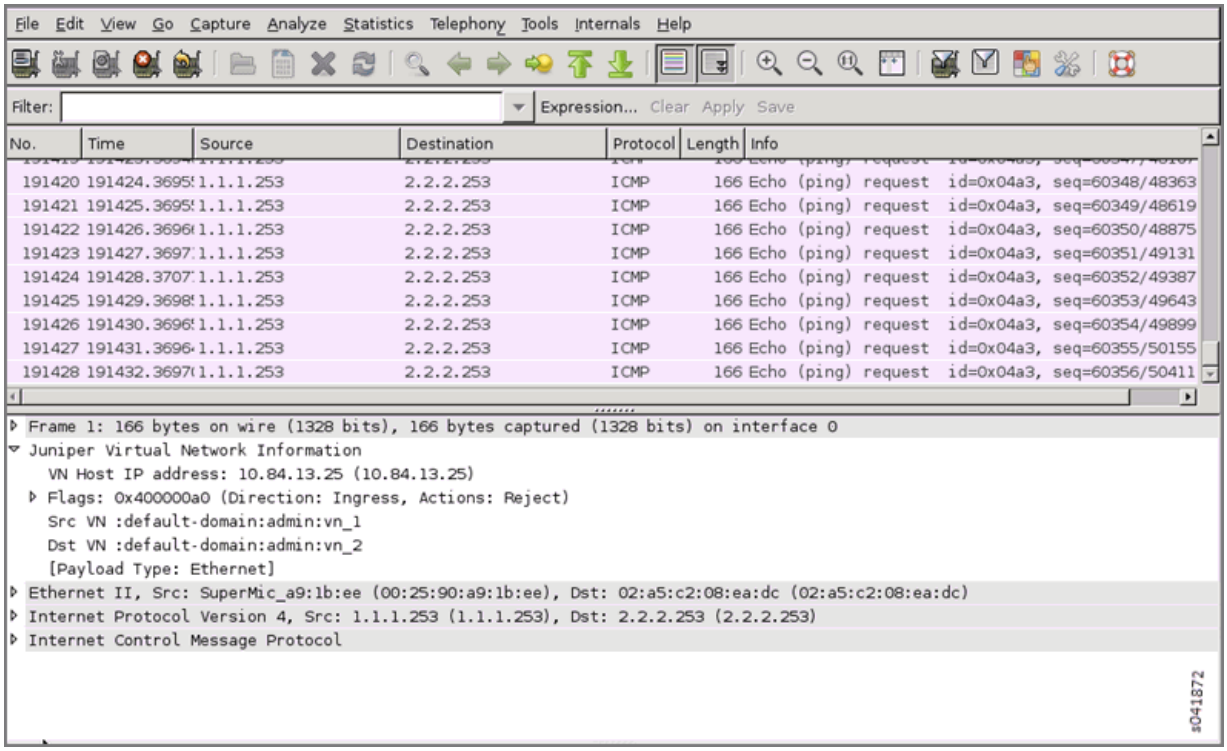
TLV end

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 27 on page 157 .

Figure 27: 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 | 144
Configuring Interface Monitoring and Mirroring | 152
Mirroring Enhancements | 153
Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 163

Using the Wireshark Plugin to Analyze Packets Between vRouter and vRouter Agent on pkt0 Interface

Wireshark is a an application that analyzes packets from a network and displays the packet information in detail.

Contrail Networking Release 2008 and later supports the Wireshark **agent_header.lua** plugin, which enables you to capture and analyze the packets exchanged between a vRouter data plane and vRouter agent. You can capture the packets by executing the `vifdump -i 2` and the `tcpdump -i pkt0` commands in DPDK mode and kernel mode respectively. In release 2008, the Wireshark agent_header.lua plugin is supported on Macintosh OS computers only. .Starting from release 2011, the Wireshark **agent_header.lua** plugin is supported on Macintosh OS as well as Windows OS computers. Wireshark also enables you to add agent header information to the captured packets.

Before you begin

You must ensure that the Wireshark application is installed on your computer. You can download Wireshark from the Download Wireshark page.

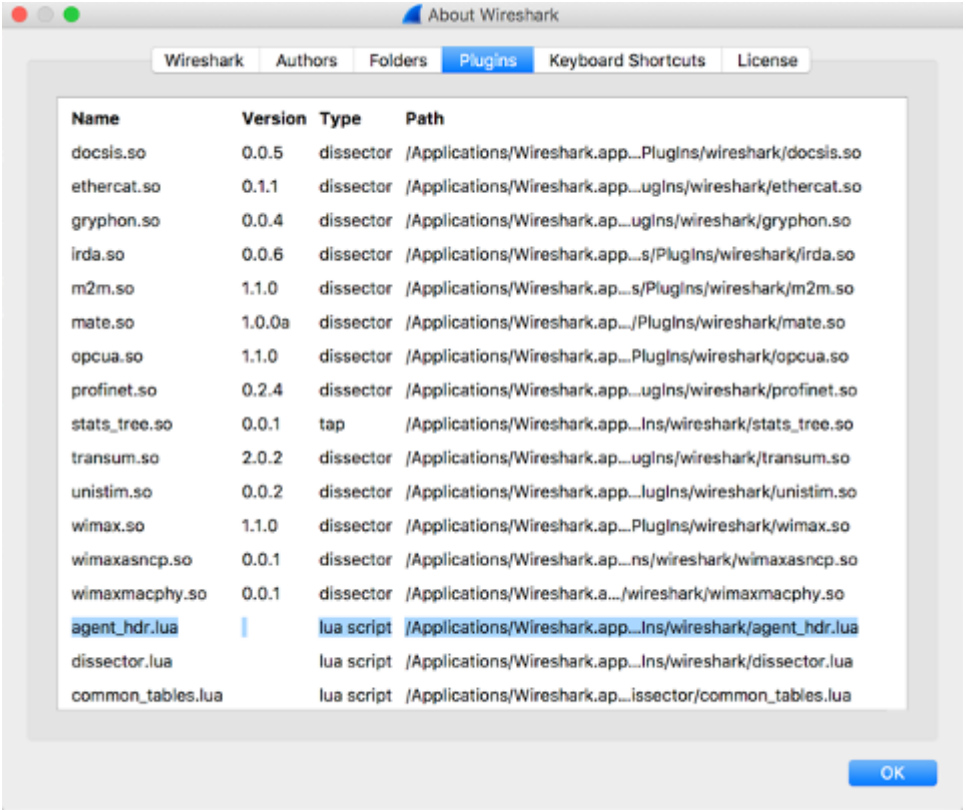
Configuration

Follow these steps to configure the Wireshark plugin and dissect agent header information in a packet:

1. Download the Wireshark plugin from GitHub: https://github.com/tungstenfabric/tf-vrouter/tree/master/utils/agent_hdr_plugin.
2. Copy the plugin in to the following Wireshark directory on your Macintosh OS computer: **/Applications/Wireshark.app/Contents/PlugIns/wireshark/**.

3. Verify that the **agent_hdr.lua** plugin is loaded successfully in Wireshark. Relaunch Wireshark and navigate to **Wireshark** > **About Wireshark** > **Plugins** to verify that the plugin is loaded in the **Plugins** section. See Figure 28 on page 159 .

Figure 28: The Plugin is Loaded in Wireshark

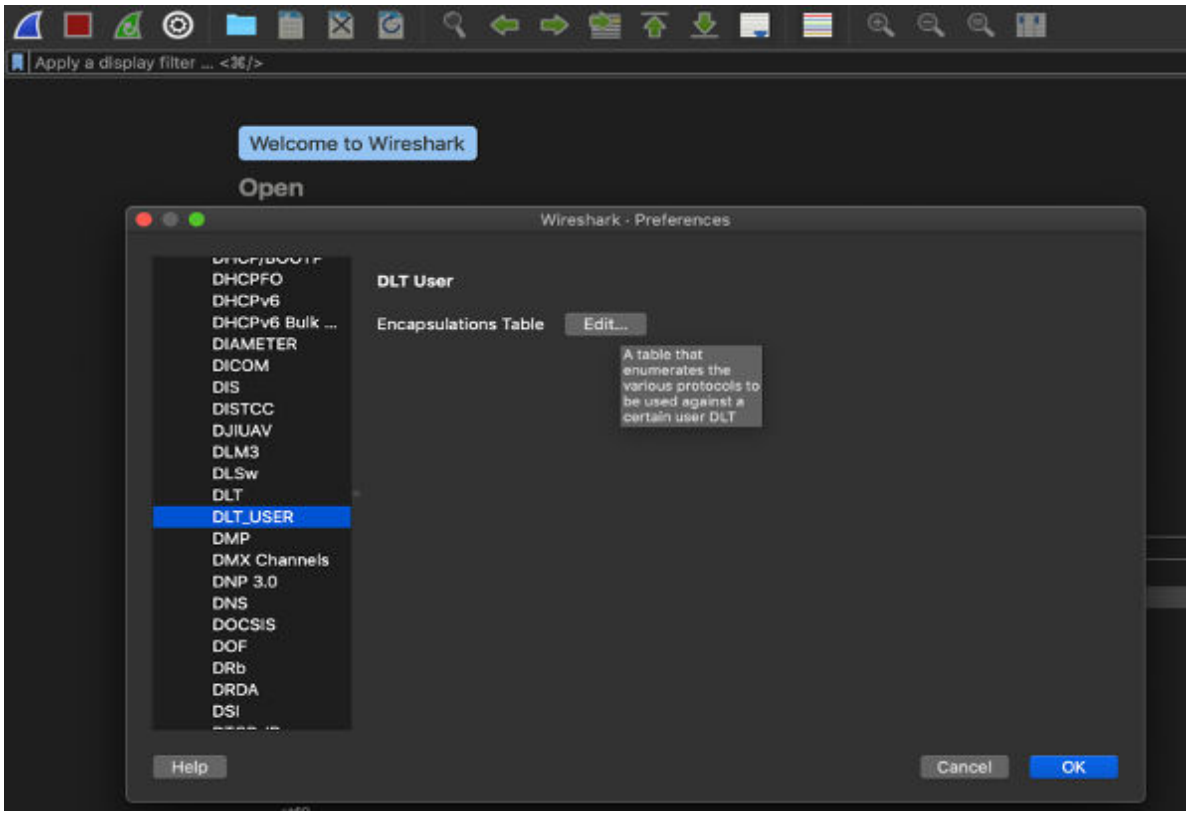


4. Pass the pcap file through editcap to add a custom encapsulation type for a packet:

```
editcap -T user0 <pcap-file-to-be-read> <output.pcap>
```

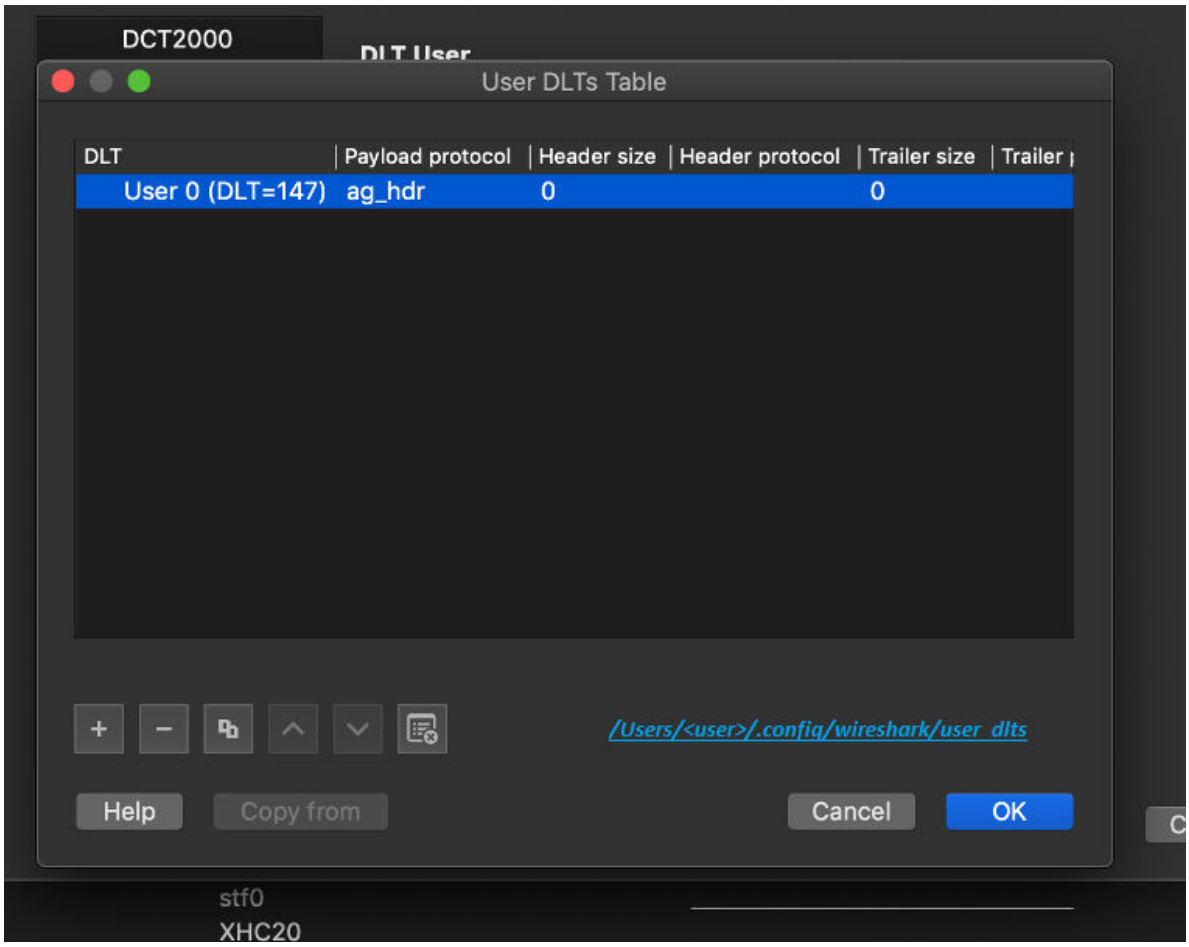
5. In Wireshark, navigate to **Wireshark** > **Preferences** > **Protocols** > **DLT_USER** > **Edit Encapsulation Table**. See Figure 29 on page 160 .

Figure 29: Edit Encapsulation Table



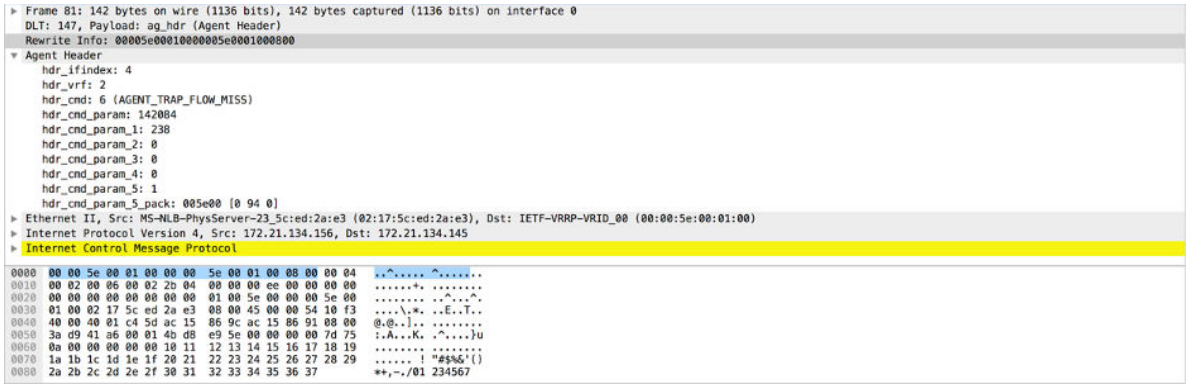
6. In the **Edit Encapsulation Table**, add the `agent_hdr` as a payload protocol for the packet. See Figure 30 on page 161 .

Figure 30: Add Agent Header to a Packet



7. Using Wireshark, open the modified pcap file you generated in step 4. Wireshark displays the parsed packets. See Figure 31 on page 161 .

Figure 31: Packets Expanded Using the Wireshark Plugin



Follow these steps to configure the Wireshark plugin in a Windows OS computer and dissect agent header information in a packet:

1. Download the Wireshark plugin from GitHub: https://github.com/tungstenfabric/tf-vrouter/tree/master/utils/agent_hdr_plugin.
2. If you are using Windows 32-bit OS, copy the plugin in to the following Wireshark directory on your computer: **C:\Program Files (x86)\Wireshark**.

   If you are using Windows 64-bit OS, copy the plugin in to the following Wireshark directory on your computer: **C:\Program Files\Wireshark\plugins**.
3. Verify that the **agent_hdr.lua** plugin is loaded successfully in Wireshark. Relaunch Wireshark and navigate to **Help** > **About Wireshark** > **Plugins** to verify that the plugin is loaded in the **Plugins** section.
4. Open command prompt in **Run as administrator** mode and navigate to **C:\Program Files\Wireshark** to use editcap. Pass the pcap file through editcap to add a custom encapsulation type for a packet:

   ```
   editcap -T user0 <pcap-file-to-be-read> <output.pcap>
   ```
5. In Wireshark, navigate to **Edit** > **Preferences** > **Protocols** > **DLT_USER** > **Edit Encapsulation Table**.
6. In the **Edit Encapsulation Table**, add the `agent_hdr` as a payload protocol for the packet. See .
7. Using Wireshark, open the modified pcap file you generated in step "4" on page 162 . Wireshark displays the parsed packets.

The **agent_header.lua** plugin is also available in contrail-tools container. You must perform the following steps to use the plugin from the contrail-tools container:

1. Log in to vRouter as a root user.
2. Use the following command to view the summary of eachpacket in the pcap file:

   ```
   tshark3_2 -nr <pcap file> -o "uat:user_dlts:\"User 0(DLT=147)\",\"ag_hdr\",\"0\",\"\",\"0\",\"\"" -t ad
   ```
3. Use the following command to view detailed informationof the packets in the pcap file:

   ```
   tshark3_2 -nr <pcap file> -o "uat:user_dlts:\"User0 (DLT=147)\",\"ag_hdr\",\"0\",\"\",\"0\",\"\"" -T pdml
   ```

Change History Table

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

Release	Description
2011	Starting from release 2011, the Wireshark **agent_header.lua** plugin is supported on Macintosh OS as well as Windows OS computers.
2008	Contrail Networking Release 2008 and later supports the Wireshark **agent_header.lua** plugin, which enables you to capture and analyze the packets exchanged between a vRouter data plane and vRouter agent.

RELATED DOCUMENTATION

Using Contrail Tools | 92

Configuring Traffic Analyzers and Packet Capture for Mirroring | 144

Analyzer Service Virtual Machine | 155

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 iany 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 | **144**

Configuring Interface Monitoring and Mirroring | **152**

Mirroring Enhancements | **153**

Analyzer Service Virtual Machine | **155**

CHAPTER 4

Using Contrail Web UI to Monitor and Troubleshoot the Network

IN THIS CHAPTER

- Monitoring the System | **165**
- Monitor > Infrastructure > Dashboard | **169**
- Monitor > Infrastructure > Control Nodes | **173**
- Monitor > Infrastructure > Virtual Routers | **184**
- Monitor > Infrastructure > Analytics Nodes | **198**
- Monitor > Infrastructure > Config Nodes | **206**
- Monitor > Networking | **210**
- Query > Flows | **222**
- Query > Logs | **232**
- Debugging Processes Using the Contrail Introspect Feature | **239**
- Example: Debugging Connectivity Using Monitoring for Troubleshooting | **244**
- Contrail Analytics Optional Modules | **251**

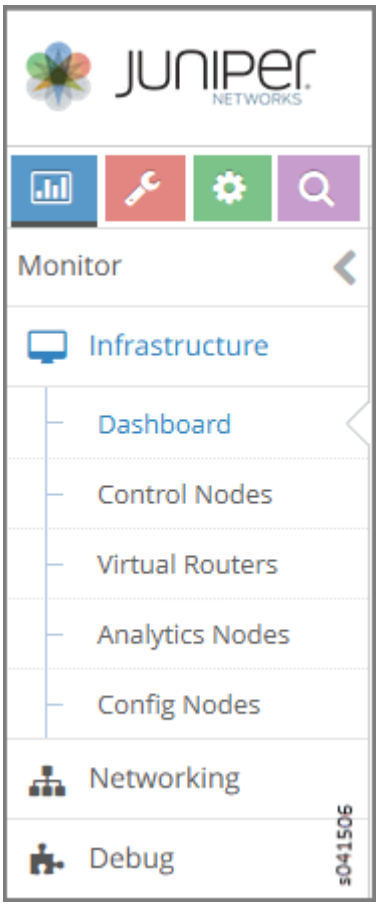
Monitoring the System

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 32 on page 166

Figure 32: Monitor Menu



See Table 16 on page 166 for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 16: MonitorMenu 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 169 .

Table 16: MonitorMenu Options *(Continued)*

Option	Description
Infrastructure > Control Nodes	View a summary for all control nodes in the system, and for each control node, view: • 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 173 .
Infrastructure > Virtual Routers	View a summary of all vRouters in the system, and for each vRouter, view: • 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 184 .
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 198 .
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 206 .

Table 16: MonitorMenu Options *(Continued)*

Option	Description
Networking > Networks	For all virtual networks for all projects in the system, view graphical traffic statistics, including: • 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. See "Monitor > Networking" on page 210 .
Networking > Dashboard	For all virtual networks for all projects in the system, view graphical traffic statistics, including: • Total traffic in and out. • Inter VN traffic in and out. 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 210 .
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.

Table 16: MonitorMenu Options *(Continued)*

Option	Description
Debug > Packet Capture	• 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 | **169**

Monitor > Infrastructure > Control Nodes | **173**

Monitor > Infrastructure > Virtual Routers | **184**

Monitor > Networking | **210**

Query > Logs | **232**

Query > Flows | **222**

Monitor > Infrastructure > Dashboard

IN THIS SECTION

- Monitor Dashboard | **170**
- Monitor Individual Details from the Dashboard | **170**
- Using Bubble Charts | **171**
- Color-Coding of Bubble Charts | **172**

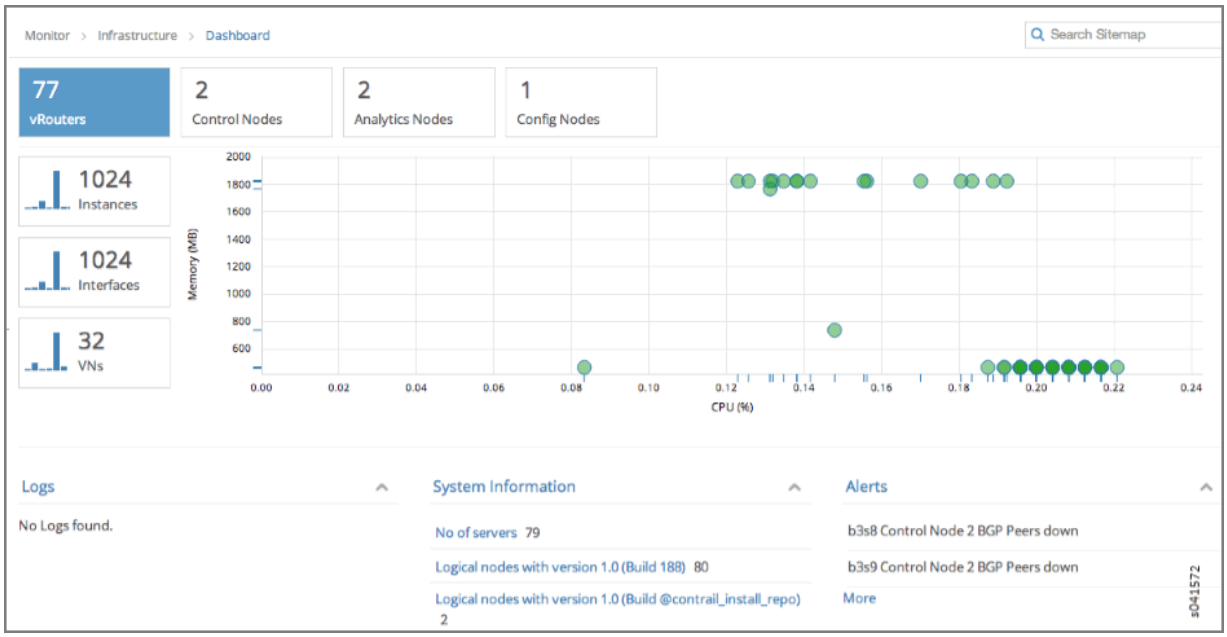
Use **Monitor > Infrastructure > Dashboard** to get an “at-a-glance” view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes

currently operational, a bubble chart of virtualrouters 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 33 on page 170 .

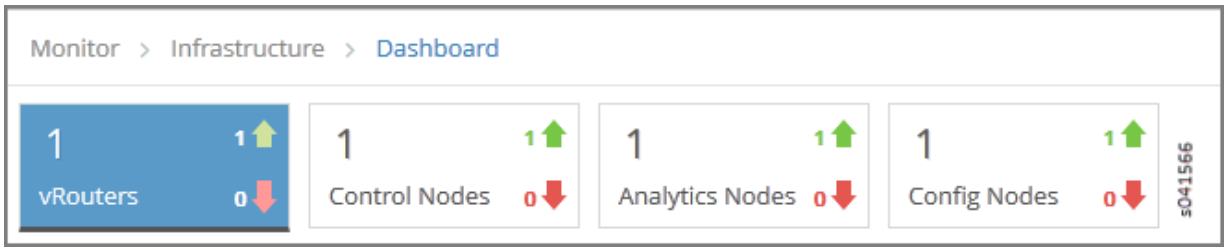
Figure 33: Monitor > Infrastructure > Dashboard



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 34 on page 171 . 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 34: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in Table 17 on page 171 .

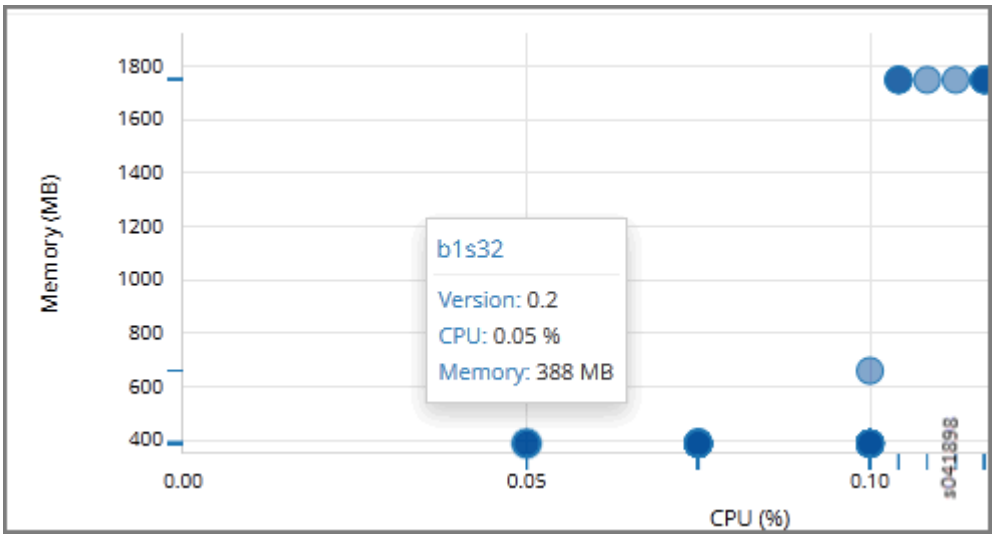
Table 17: Dashboard Summary Boxes

Box	For More Information
vRouters	"Monitor > Infrastructure > Virtual Routers" on page 184
Control Nodes	"Monitor > Infrastructure > Control Nodes" on page 173
Analytics Nodes	"Monitor > Infrastructure > Analytics Nodes" on page 198
Config Nodes	"Monitor > Infrastructure > Config Nodes" on page 206

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 likeso on. You can hover over any bubble to get summary information about the component it represents; see Figure 35 on page 172 . You can click through the summary information to get more details about the component.

Figure 35: 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 | **184**

Monitor > Infrastructure > Control Nodes | **173**

Monitor > Infrastructure > Analytics Nodes | **198**

Monitor > Infrastructure > Config Nodes | **206**

Monitor > Infrastructure > Control Nodes

IN THIS SECTION

- Monitor Control Nodes Summary | 173
- Monitor Individual Control Node Details | 174
- Monitor Individual Control Node Console | 176
- Monitor Individual Control Node Peers | 179
- Monitor Individual Control Node Routes | 181

Navigate to **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 36 on page 173 . See Table 18 on page 174 for descriptions of the fields on this screen.

Figure 36: Control Nodes Summary

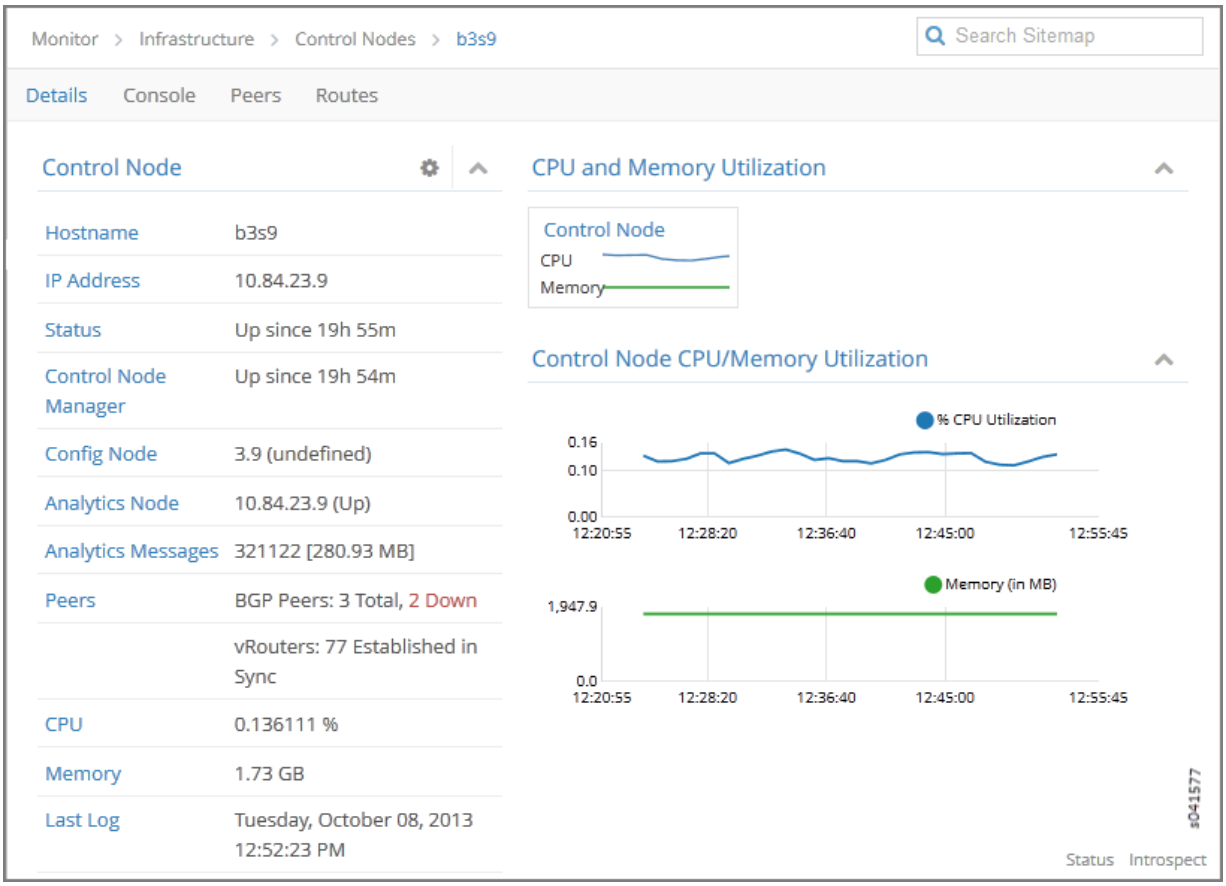
Table 18: Control Nodes Summary Fields

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** titleto 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 37 on page 175 .

Figure 37: 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 19 on page 175 for descriptions of the fields on this tab.

Table 19: 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.
Control Node Manager	The operational status of the control node manager.

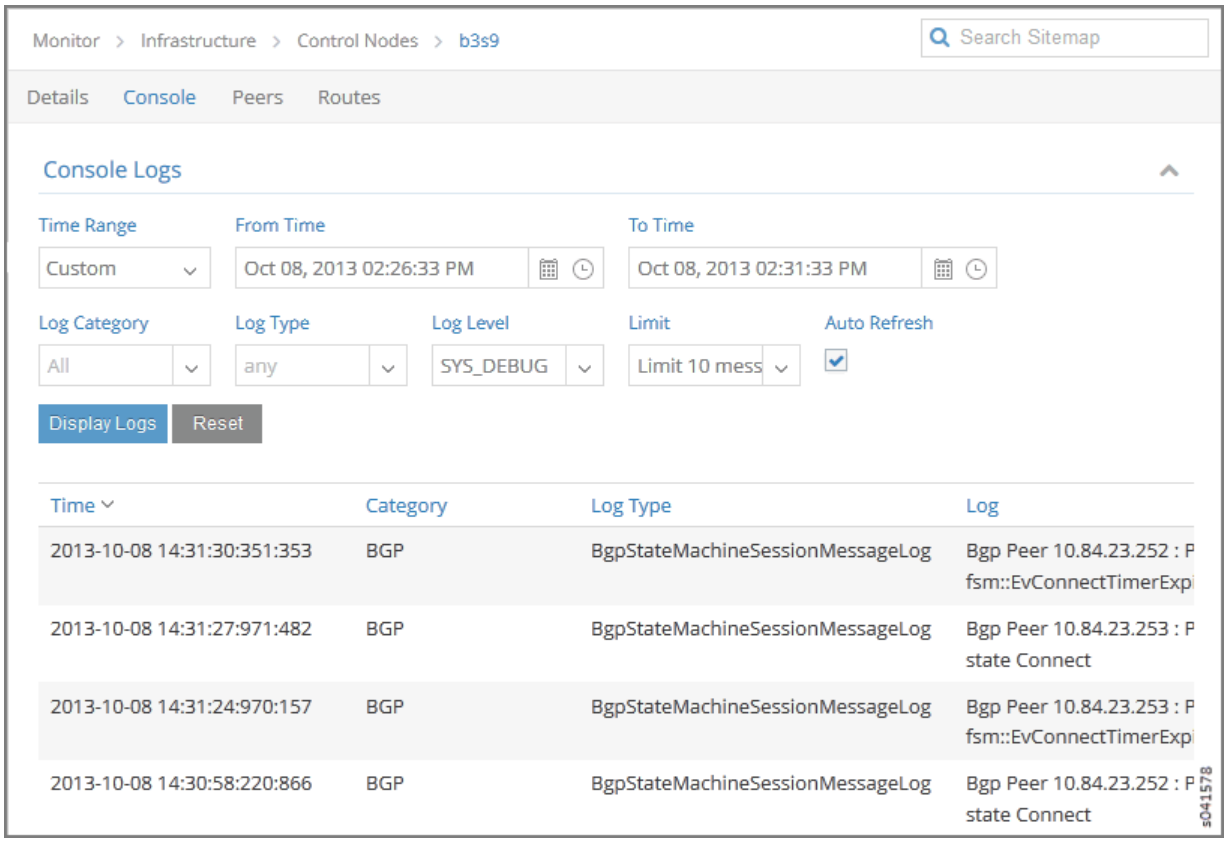
Table 19: Individual Control Node—Details Tab Fields *(Continued)*

Field	Description
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 38 on page 177 .

Figure 38: Individual Control Node—Console Tab



See Table 20 on page 177 for descriptions of the fields on the **Console** tab screen.

Table 20: 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: 1. All 2. _default_ 3. XMPP 4. TCP

Table 20: Control Node: Console Tab Fields *(Continued)*

Field	Description
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: 1. SYS_EMERG 2. SYS_ALERT 3. SYS_CRIT 4. SYS_ERR 5. SYS_WARN 6. SYS_NOTICE 7. SYS_INFO 8. SYS_DEBUG
Search	Enter any text string to search and display logs containing that string.
Limit	Select from a list an amount to limit the number of messages displayed: 1. No Limit 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. 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.

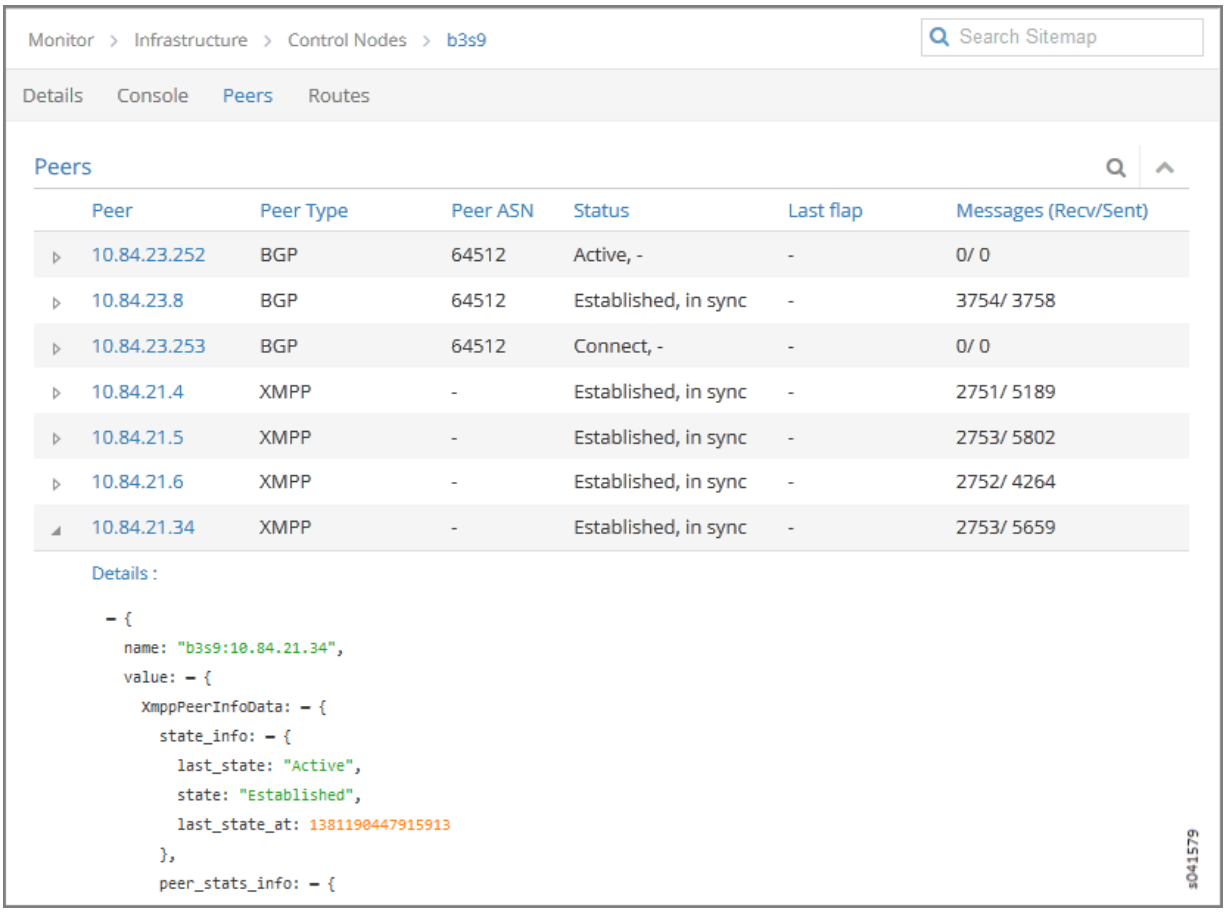
Table 20: Control Node: Console Tab Fields *(Continued)*

Field	Description
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
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 39 on page 180 .

Figure 39: Individual Control Node—Peers Tab



See Table 21 on page 180 for descriptions of the fields on the **Peers** tab screen.

Table 21: 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.

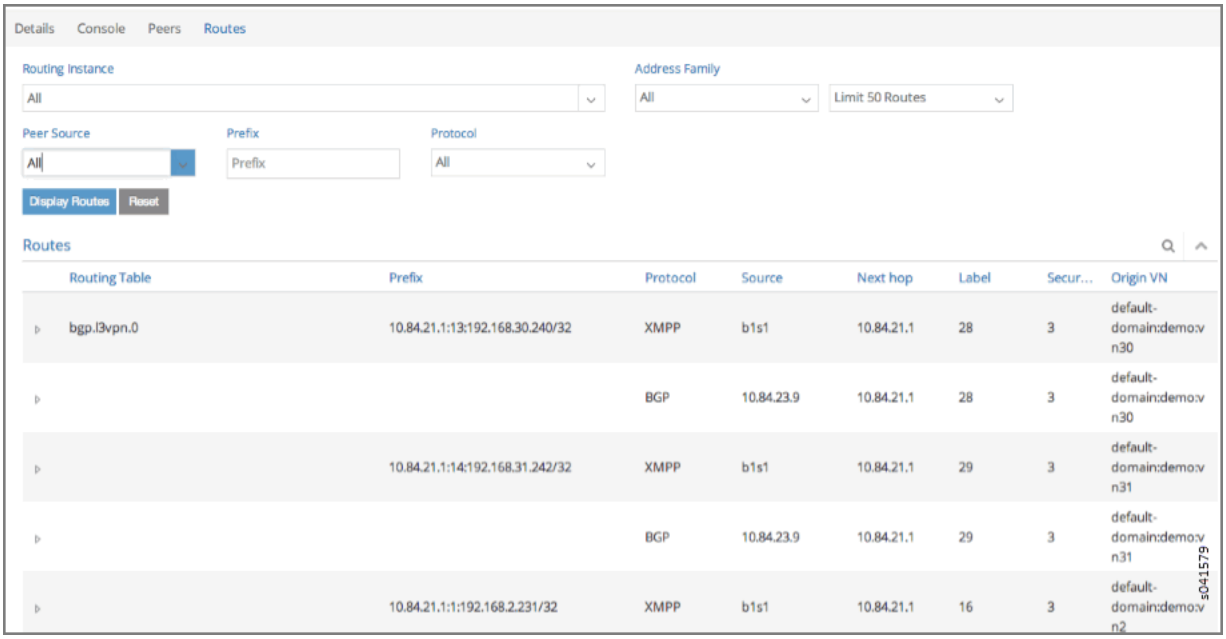
Table 21: Control Node: Peers Tab Fields *(Continued)*

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 40 on page 181 .

Figure 40: Individual Control Node—Routes Tab



See Table 22 on page 182 for descriptions of the fields on the **Routes** tab screen.

Table 22: 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.
Address Family	Select an address family for which to display the active routes: 1. All (default) 2. l3vpn 3. inet 4. inetmcast
(Limit Field)	Select to limit the display of active routes: 1. Limit 10 Routes 2. Limit 50 Routes 3. Limit 100 Routes 4. 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: 1. All (default) 2. XMPP 3. BGP 4. ServiceChain 5. Static

Table 22: Control Node: Routes Tab Fields *(Continued)*

Field	Description
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.
Column	*Description*
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.
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

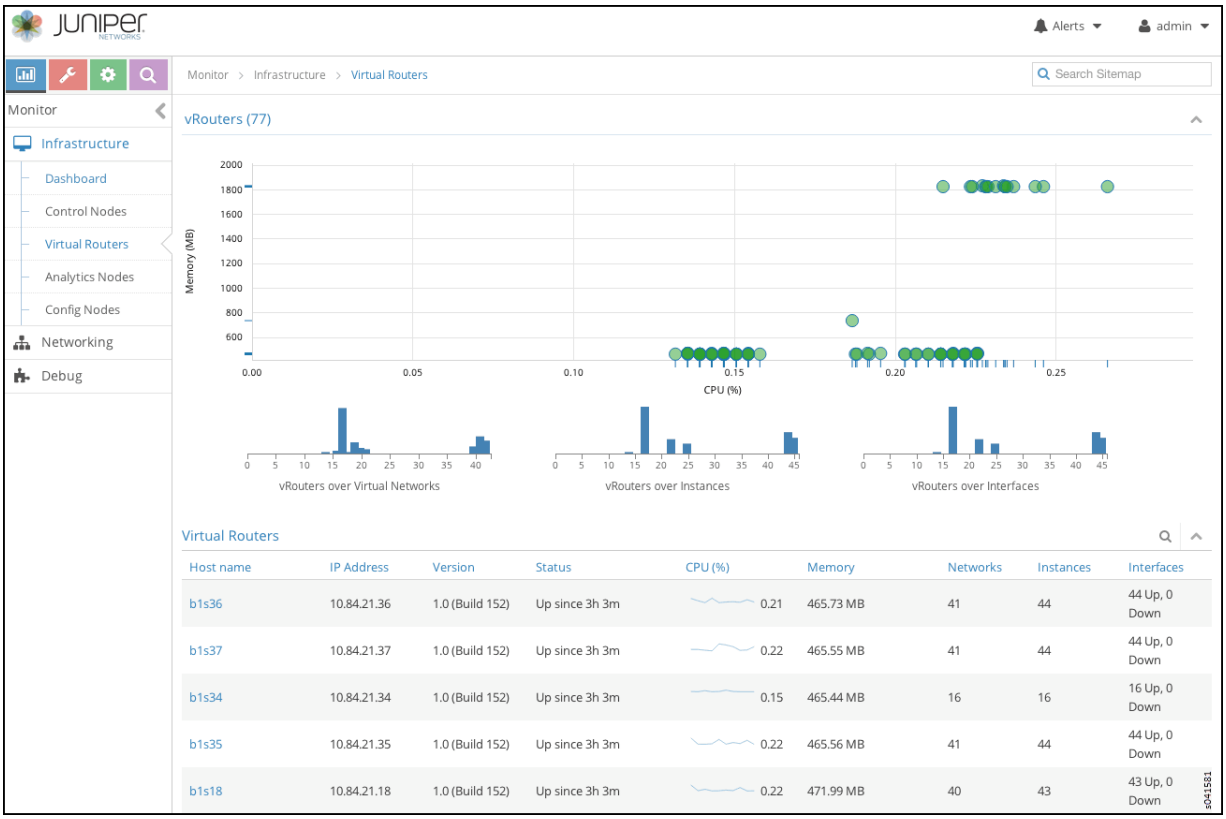
IN THIS SECTION

- Monitor vRouters Summary | 184
- Monitor Individual vRouters Tabs | 186
- Monitor Individual vRouter Details Tab | 186
- Monitor Individual vRouters Interfaces Tab | 188
- Monitor Individual vRouters Networks Tab | 190
- Monitor Individual vRouters ACL Tab | 191
- Monitor Individual vRouters Flows Tab | 193
- Monitor Individual vRouters Routes Tab | 194
- Monitor Individual vRouter Console Tab | 195

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See Figure 41 on page 185 .

Figure 41: vRouters Summary



See Table 23 on page 185 for descriptions of the fields on the **vRouters Summary** screen.

Table 23: 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.

Table 23: vRouters Summary Fields *(Continued)*

Field	Description
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.
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 42 on page 187 .
- **Console**—similar display of information as on individual control nodes **Console** tab. See Figure 48 on page 196 .
- **Interfaces**—details about associated interfaces. See Figure 43 on page 189 .
- **Networks**—details about associated networks. See Figure 44 on page 190 .
- **ACL**—details about access control lists. See Figure 45 on page 192 .
- **Flows**—details about associated traffic flows. See Figure 46 on page 193 .
- **Routes**—details about associated routes. See Figure 47 on page 195 .

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 42 on page 187 . SeeTable 24 on page 187 for descriptions of the fields on this tab.

Figure 42: Individual vRouters—Details Tab

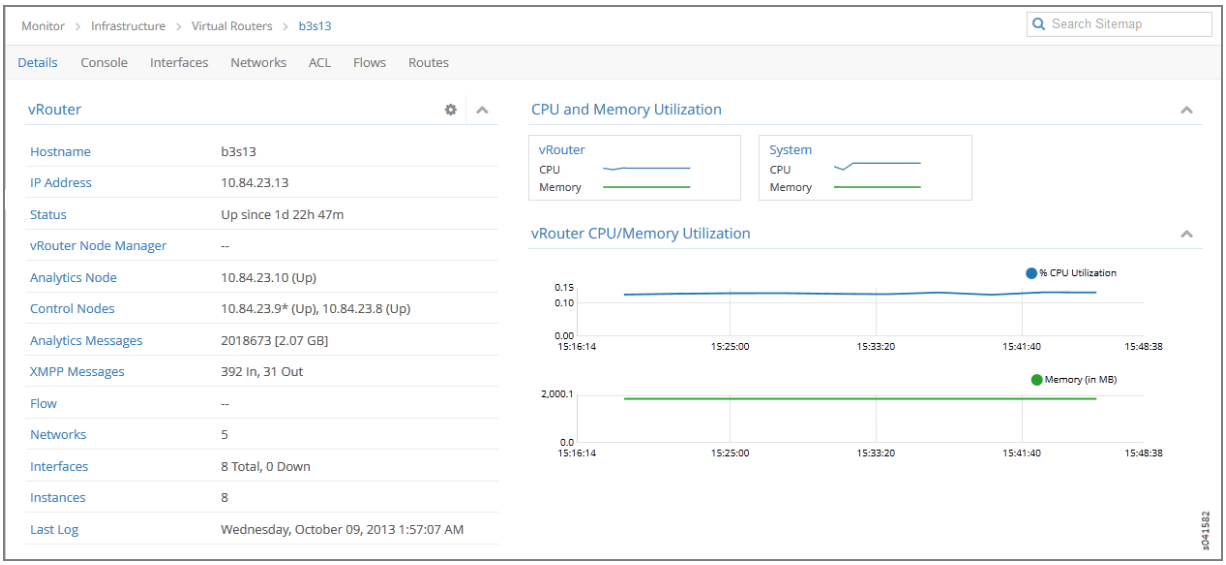


Table 24: vRouters Details Tab Fields

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.

Table 24: vRouters Details Tab Fields *(Continued)*

Field	Description
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.
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 43 on page 189 . See Table 25 on page 189 for descriptions of the fields on the **Interfaces** tab screen.

Figure 43: Individual vRouters—Interfaces Tab

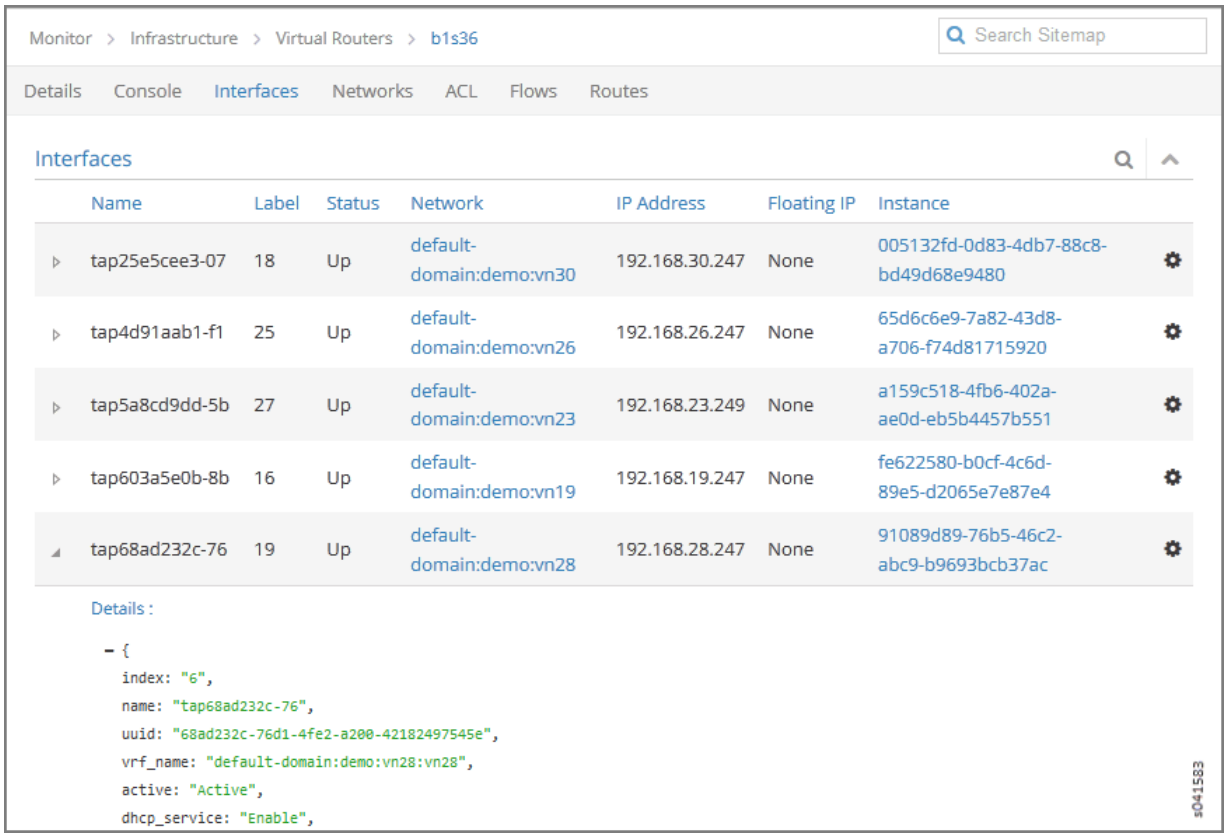


Table 25: vRouters: Interfaces Tab Fields

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.

Table 25: vRouters: Interfaces Tab Fields *(Continued)*

Field	Description
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 44 on page 190 . See Table 26 on page 191 for descriptions of the fields on the **Networks** tab screen.

Figure 44: Individual vRouters—Networks Tab

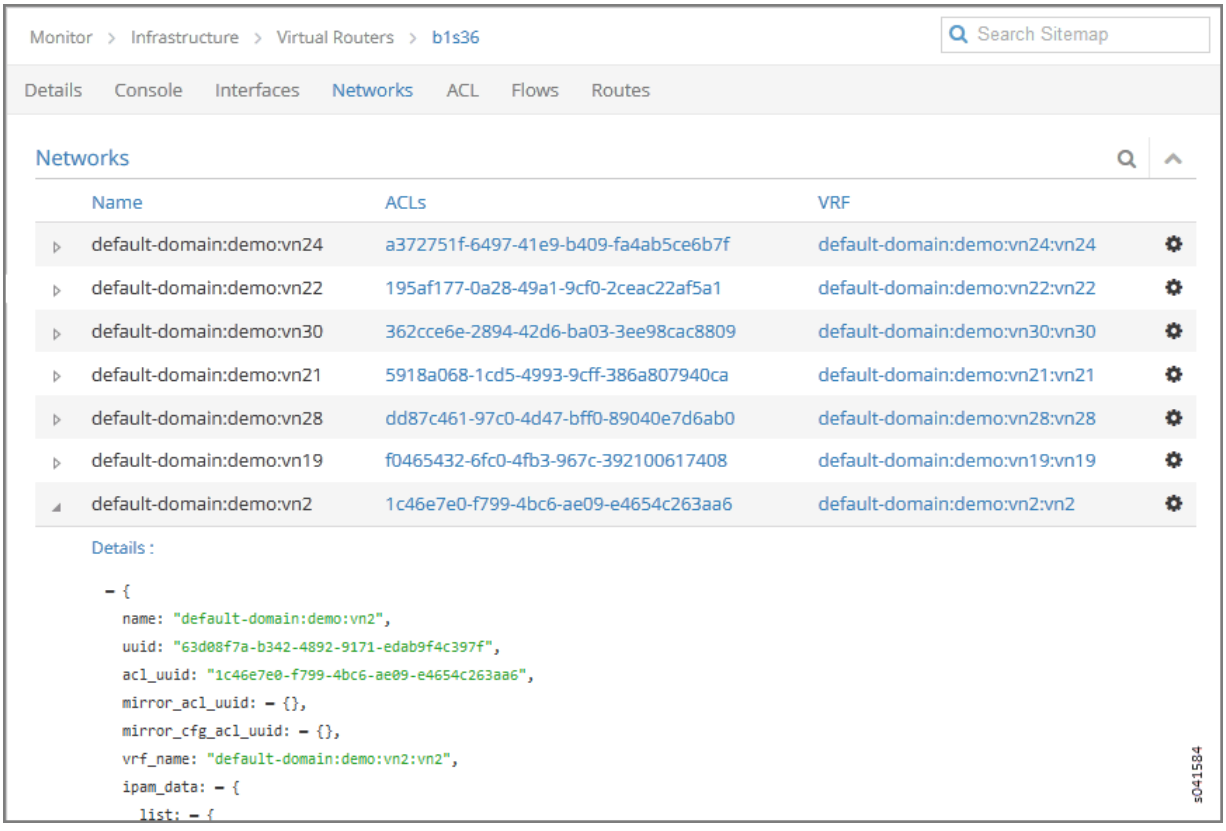


Table 26: vRouters: Networks Tab Fields

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 45 on page 192 . See Table 27 on page 192 for descriptions of the fields on the **ACL** tab screen.

Figure 45: Individual vRouters—ACL Tab

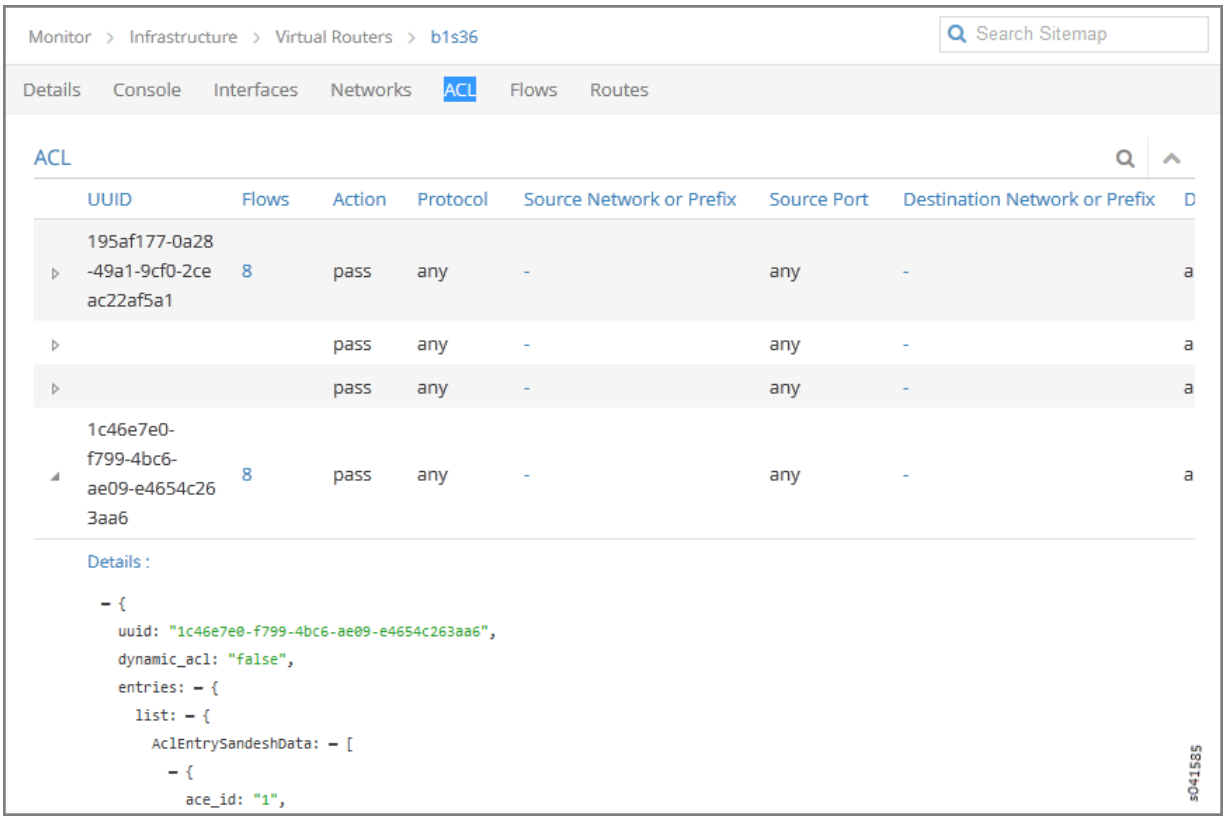


Table 27: vRouters: ACL Tab Fields

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.

Table 27: vRouters: ACL Tab Fields *(Continued)*

Field	Description
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 arrrow 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 46 on page 193 . See Table 28 on page 194 for descriptions of the fields on the **Flows** tab screen.

Figure 46: Individual vRouters—Flows Tab

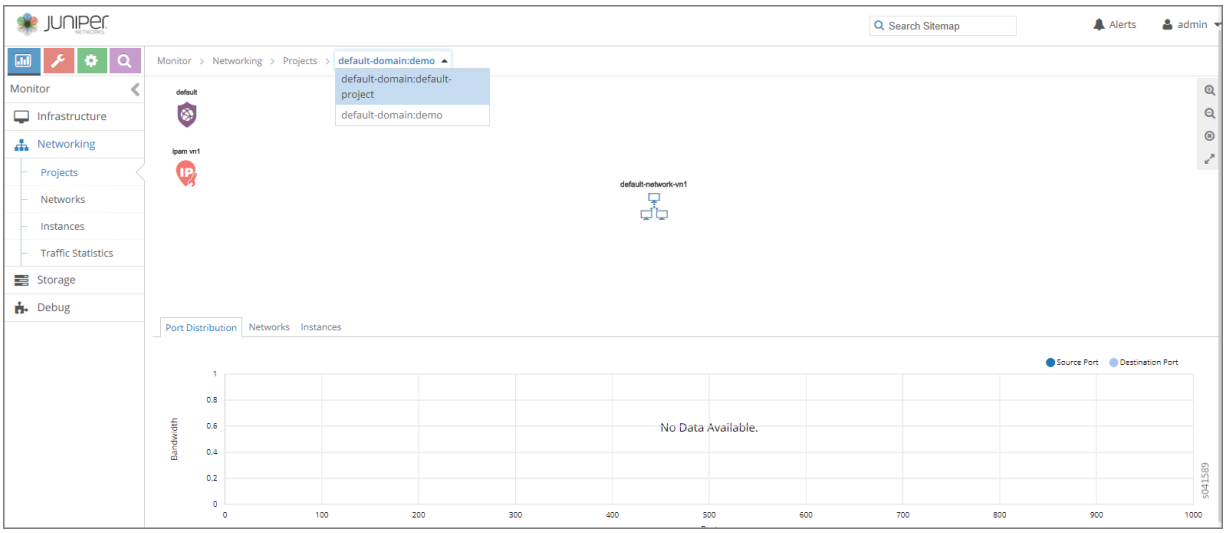


Table 28: vRouters: Flows Tab Fields

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.
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 47 on page 195 . See Table 29 on page 195 for descriptions of the fields on the **Routes** tab screen.

Figure 47: Individual vRouters—Routes Tab

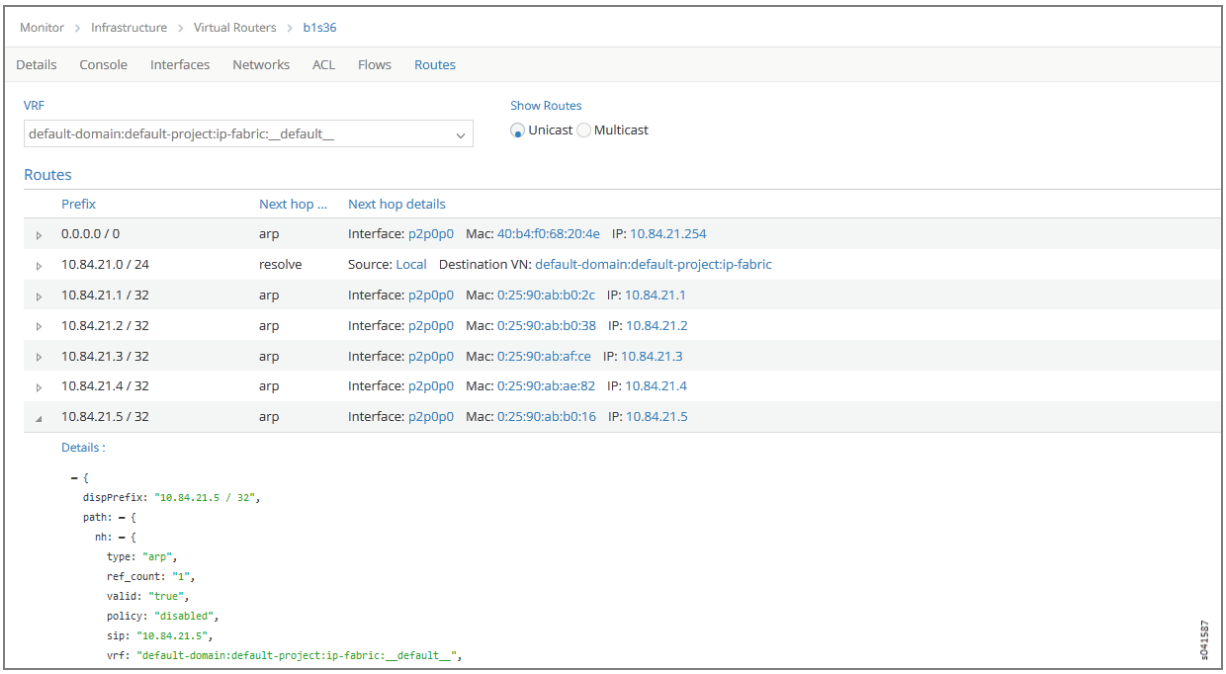


Table 29: vRouters: Routes Tab Fields

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**.
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 48 on page 196 . See Table 30 on page 196 for descriptions of the fields on the **Console** tab screen.

Figure 48: Individual vRouter—Console Tab

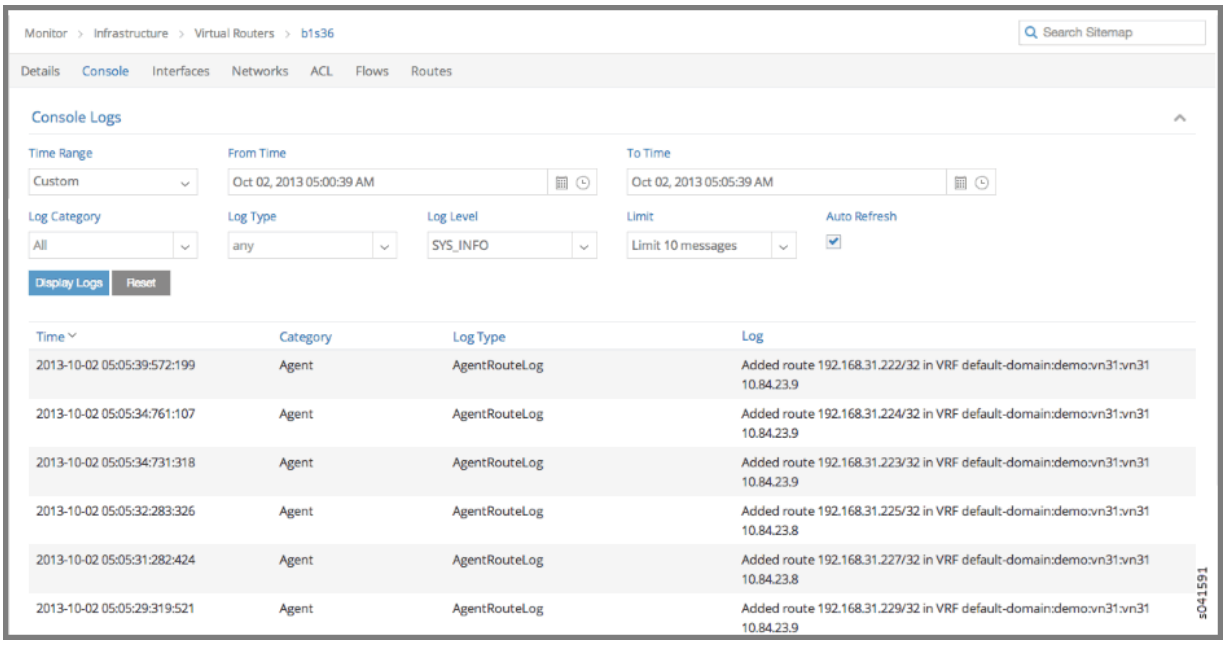


Table 30: Control Node: Console Tab Fields

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.
To Time	If you select **Custom** in **Time Range**, enter the end time.
Log Category	Select a log category to display: • All • _default_ • XMPP • TCP
Log Type	Select a log type to display.

Table 30: Control Node: Console Tab Fields *(Continued)*

Field	Description
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
Limit	Select from a list an amount to limit the number of messages displayed: • 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.
Columns	

Table 30: Control 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 > Analytics Nodes

IN THIS SECTION

- Monitor Analytics Nodes | **198**
- Monitor Analytics Individual Node Details Tab | **200**
- Monitor Analytics Individual Node Generators Tab | **201**
- Monitor Analytics Individual Node QE Queries Tab | **202**
- Monitor Analytics Individual Node Console Tab | **203**

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 49 on page 199 . See Table 31 on page 199 for descriptions of the fields on the analytics summary.

Figure 49: Analytics Nodes Summary

Table 31: 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 50 on page 200 .

See Table 32 on page 200 for descriptions of the fields on this screen.

Figure 50: Monitor Analytics Individual Node Details Tab

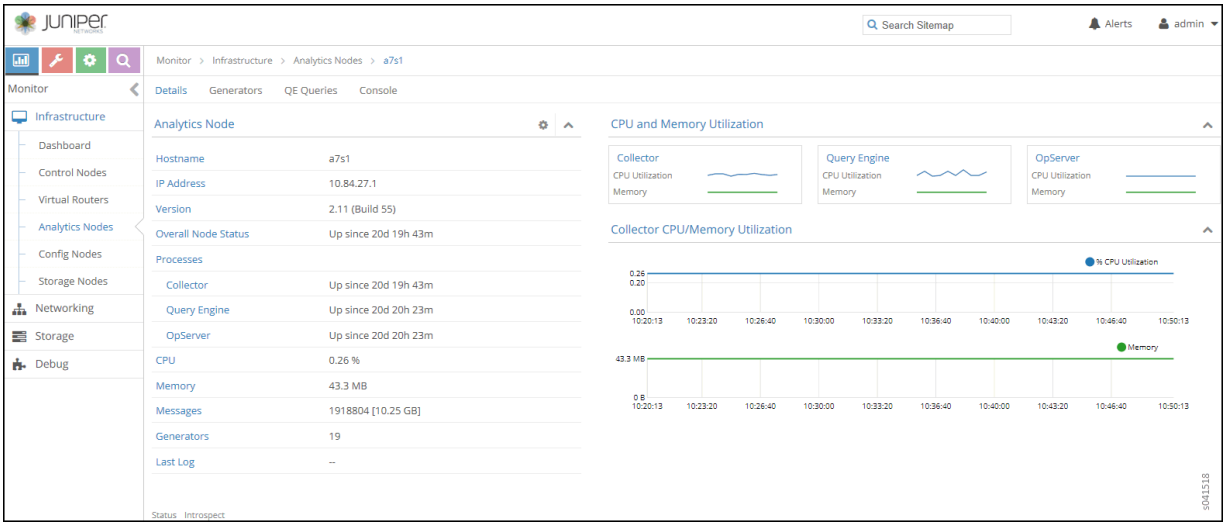


Table 32: Monitor Analytics Individual Node Details Tab Fields

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.

Table 32: Monitor Analytics Individual Node Details Tab Fields *(Continued)*

Field	Description
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 51 on page 201 . Click the expansion arrow next to any generator name to reveal more details. See Table 33 on page 202 for descriptions of the fields on the **Peers** tab screen.

Figure 51: Individual Analytics Node—Generators Tab

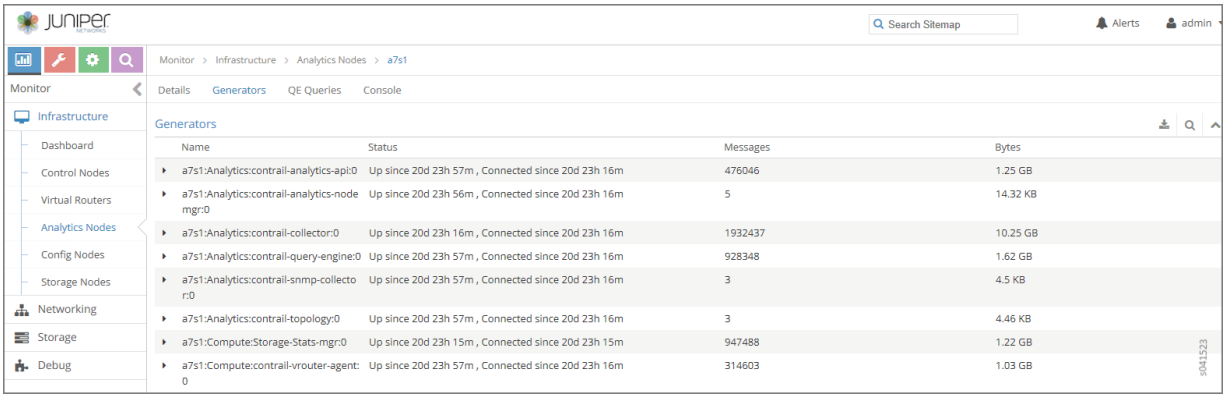


Table 33: Monitor Analytics Individual Node Generators Tab Fields

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 52 on page 202 .

See Table 34 on page 202 for descriptions of the fields on the **QE Queries** tab screen.

Figure 52: Individual Analytics Node—QE QueriesTab

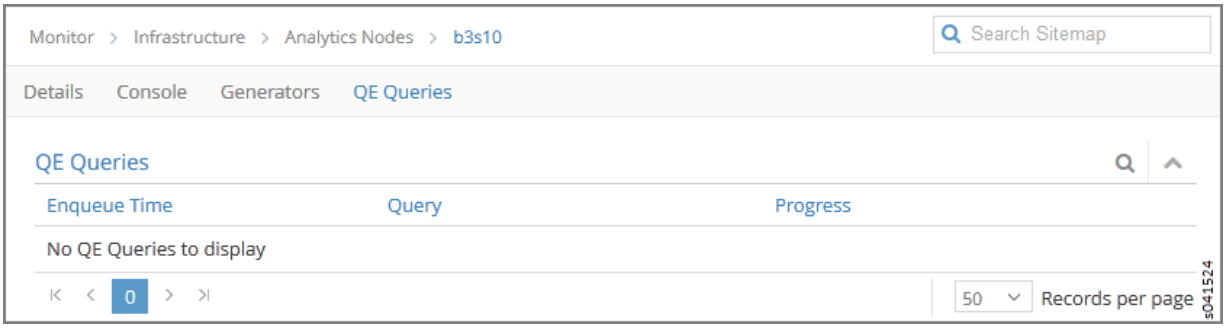


Table 34: Analytics Node QE Queries Tab Fields

Field	Description
Enqueue Time	The length of time this message has been in the queue waiting to be delivered.
Query	The query message.

Table 34: Analytics Node QE Queries Tab Fields *(Continued)*

Field	Description
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 53 on page 203 . See Table 35 on page 203 for descriptions of the fields on the **Console** tab screen.

Figure 53: Analytics Individual Node—Console Tab

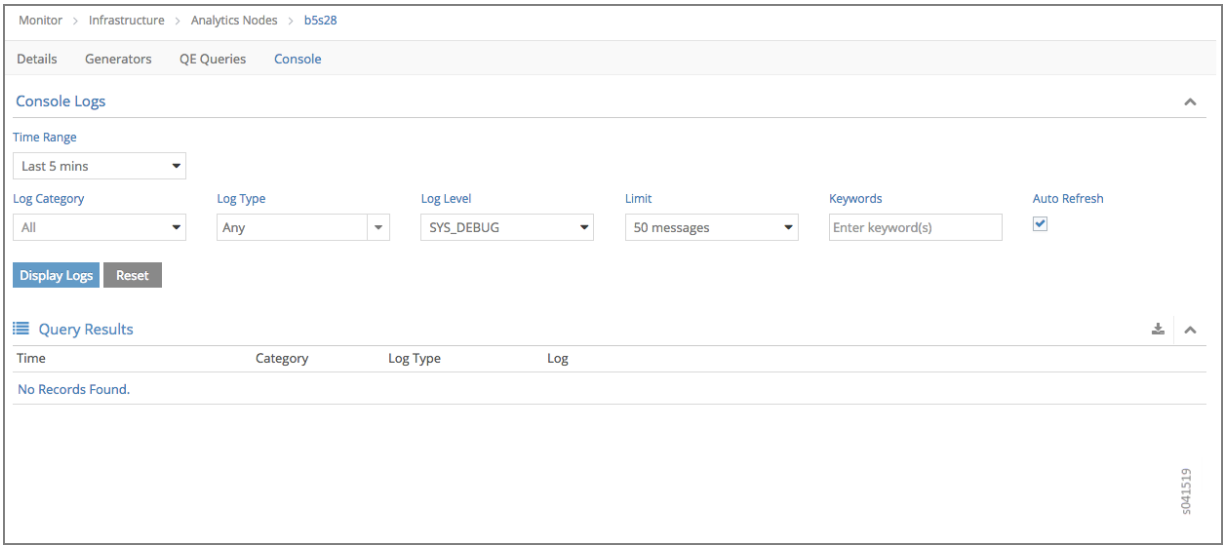


Table 35: Monitor Analytics Individual Node Console Tab Fields

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**.

Table 35: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
Log Category	Select a log category to display: 1. All 2. _default_ 3. XMPP 4. TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: 1. SYS_EMERG 2. SYS_ALERT 3. SYS_CRIT 4. SYS_ERR 5. SYS_WARN 6. SYS_NOTICE 7. SYS_INFO 8. SYS_DEBUG
Keywords	Enter any text string to search for and display logs containing that string.

Table 35: Monitor Analytics Individual Node Console Tab Fields *(Continued)*

Field	Description
(Limit field)	Select the number of messages to display: 1. No Limit 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. 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.
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 | 206
- Monitor Individual Config Node Details | 207
- Monitor Individual Config Node Console | 208

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 54 on page 206 .

Figure 54: Config Nodes Summary

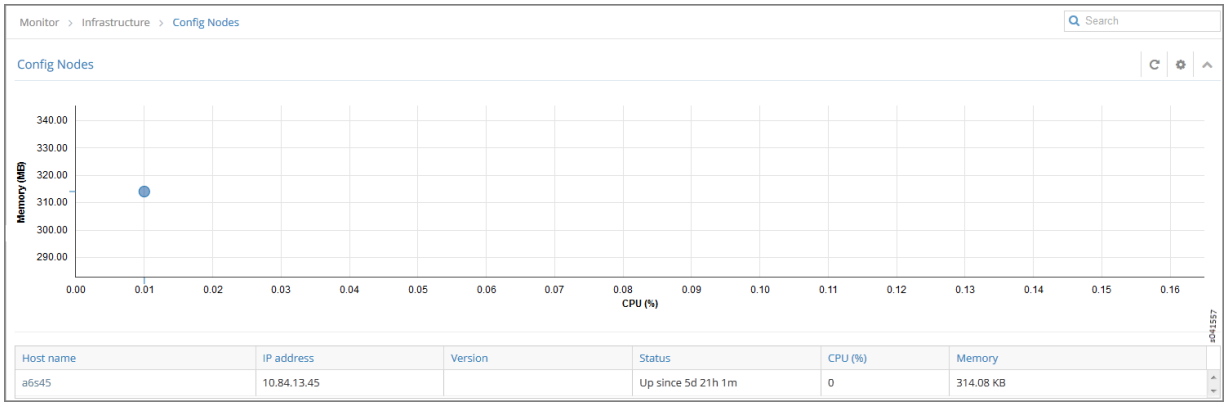


Table 36 on page 206 describes the fields in the Config Nodes summary.

Table 36: Config Nodes Summary Fields

Field	Description
Host name	The name of this node.

Table 36: Config Nodes Summary Fields *(Continued)*

Field	Description
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 55 on page 207 .

Figure 55: Individual Config Nodes— Details Tab

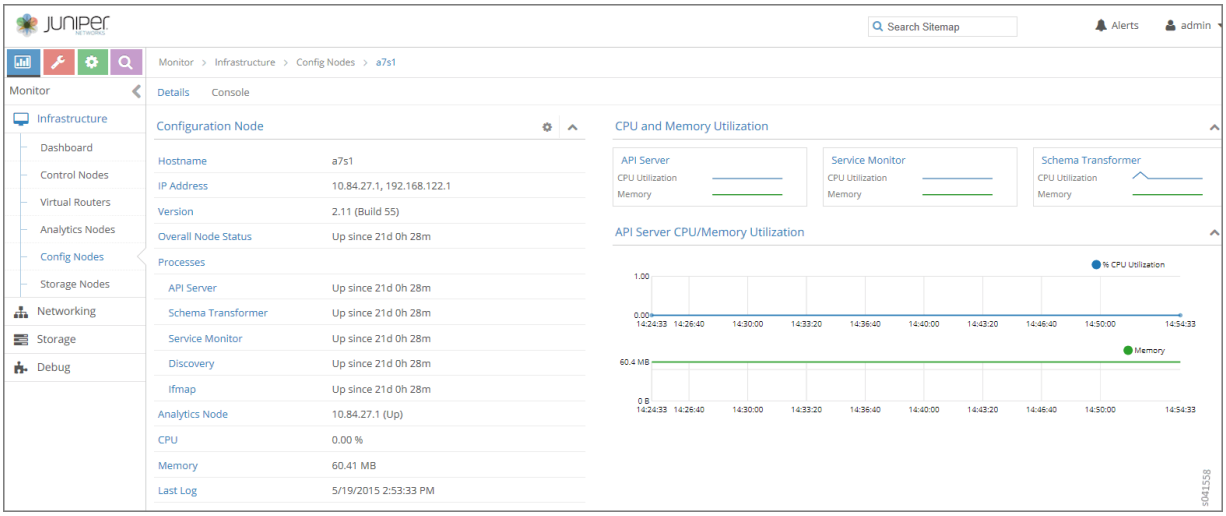


Table 37 on page 208 describes the fields on the Details screen.

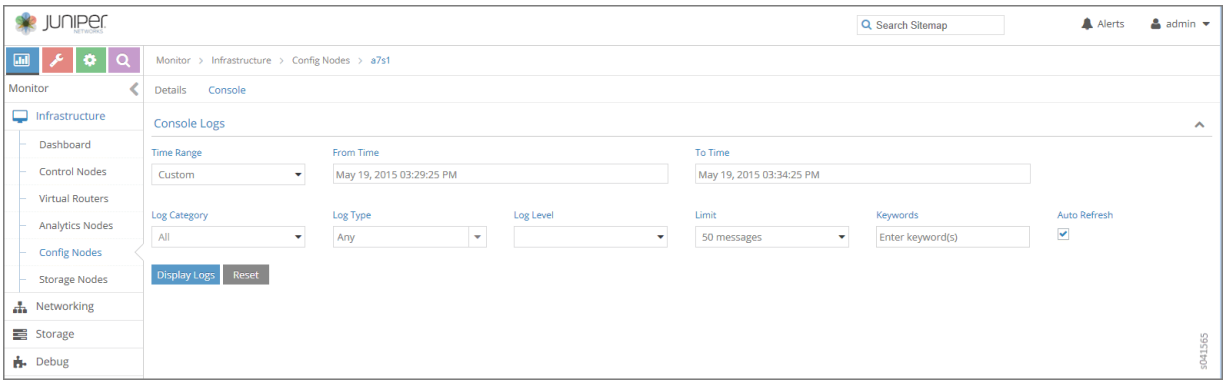
Table 37: Individual Config Nodes— Details Tab Fields

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 56 on page 209 .

Figure 56: Individual Config Node—Console Tab



See Table 38 on page 209 for descriptions of the fields on the **Console** tab screen.

Table 38: 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: 1. All 2. Limit 10 messages 3. Limit 50 messages 4. Limit 100 messages 5. Limit 200 messages 6. Limit 500 messages

Table 38: Individual Config Node-Console Tab Fields *(Continued)*

Field	Description
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.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

Monitor > Networking

IN THIS SECTION

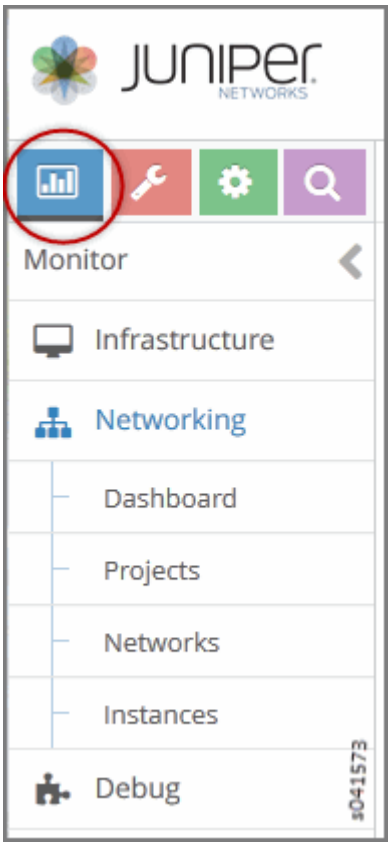
- Monitor > Networking Menu Options | **210**
- Monitor > Networking > Dashboard | **211**
- Monitor > Networking > Projects | **213**
- Monitor Projects Detail | **214**
- Monitor > Networking > Networks | **217**

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 57 on page 211 shows the menu options available under **Monitor > Networking**.

Figure 57: 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 58 on page 212 .

Figure 58: Traffic Statistics for Domain Window

Table 39 on page 212 describes the fields in the Traffic Statistics for Domain window.

Table 39: 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.

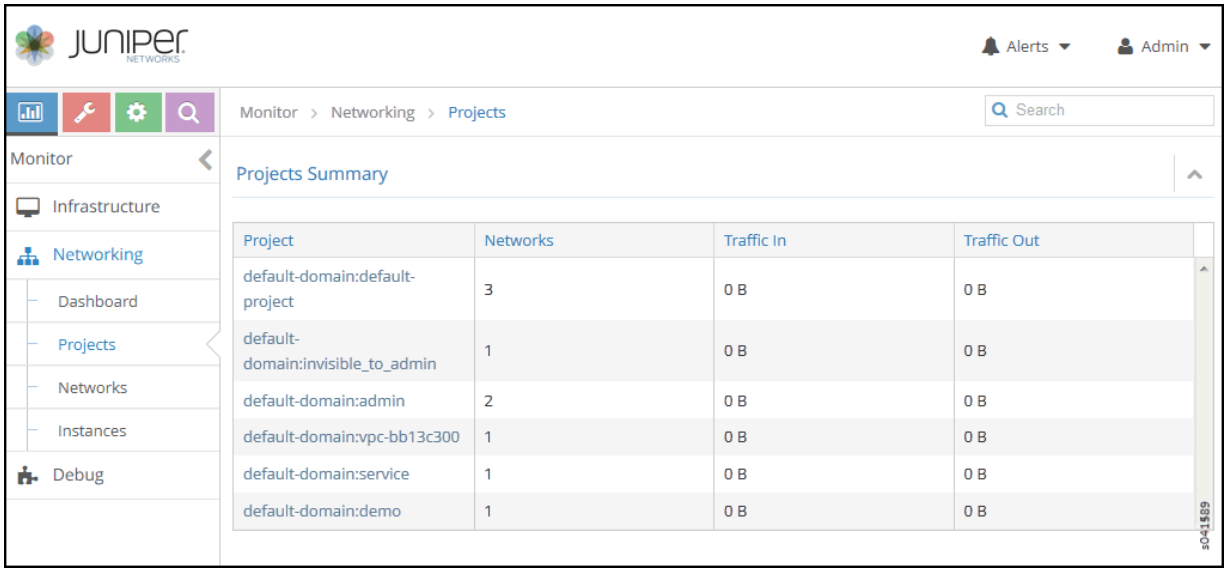
Table 39: Projects Summary Fields *(Continued)*

Field	Description
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 59 on page 213 .

Figure 59: Monitor > Networking > Projects



See Table 40 on page 214 for descriptions of the fields on this screen.

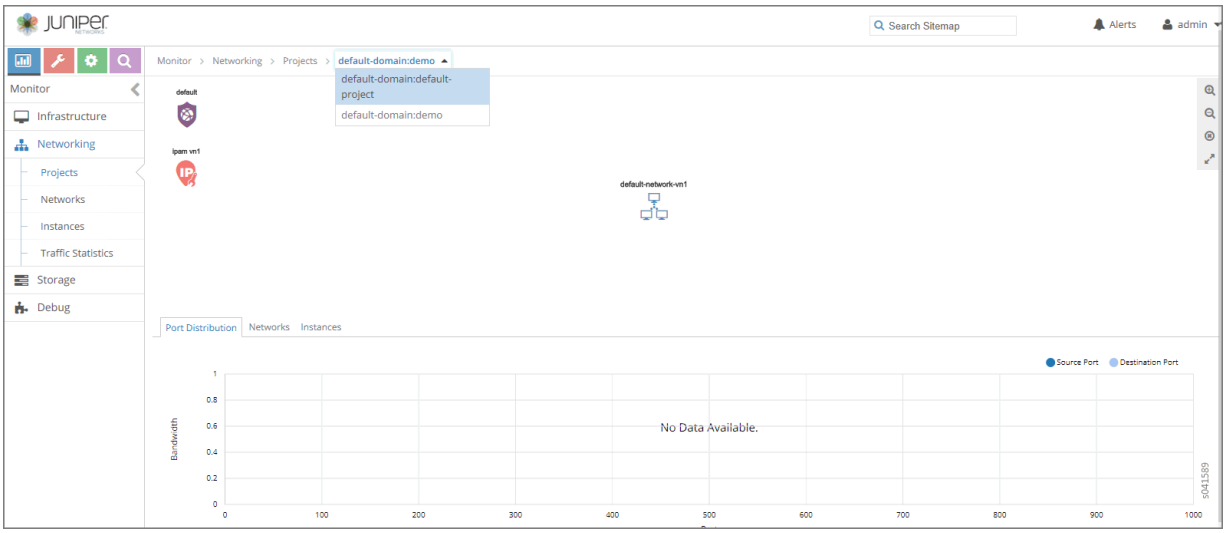
Table 40: Projects Summary Fields

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 60 on page 214 . Hover over any of the connections to get more details.

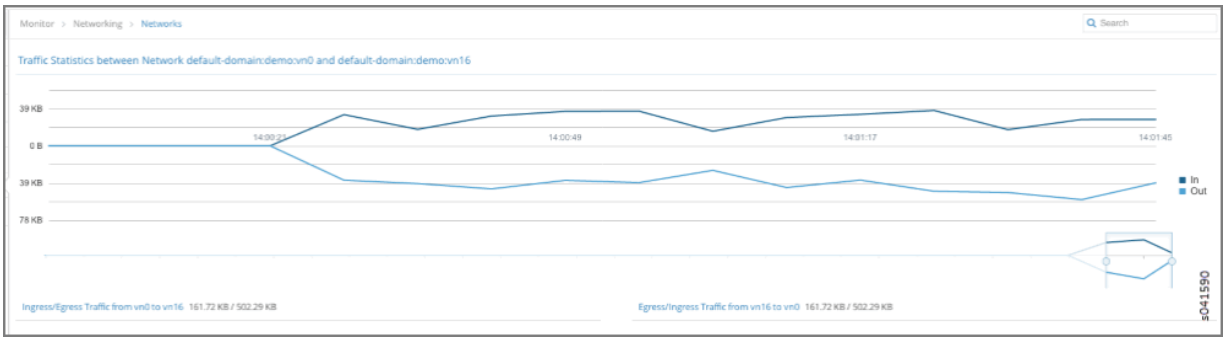
Figure 60: Monitor Projects Connectivity Details



In the Connectivity Details window you can click the links between the virtual networks to view the traffic statistics between the virtual networks.

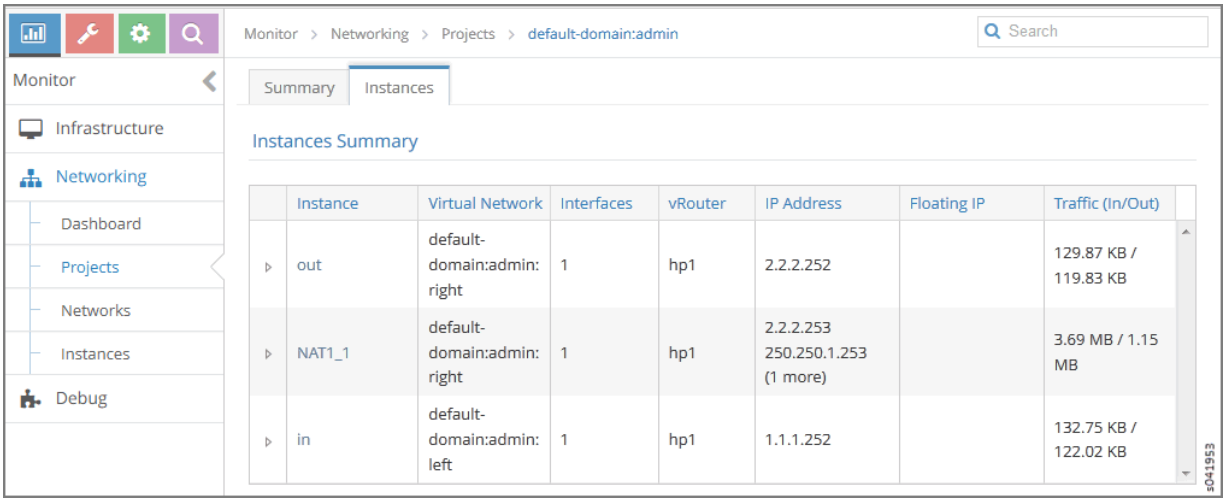
The Traffic Statistics information is also available when you select **Monitor > Networking > Networks** as shown in Figure 61 on page 215 .

Figure 61: 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 62: Projects Instances Summary



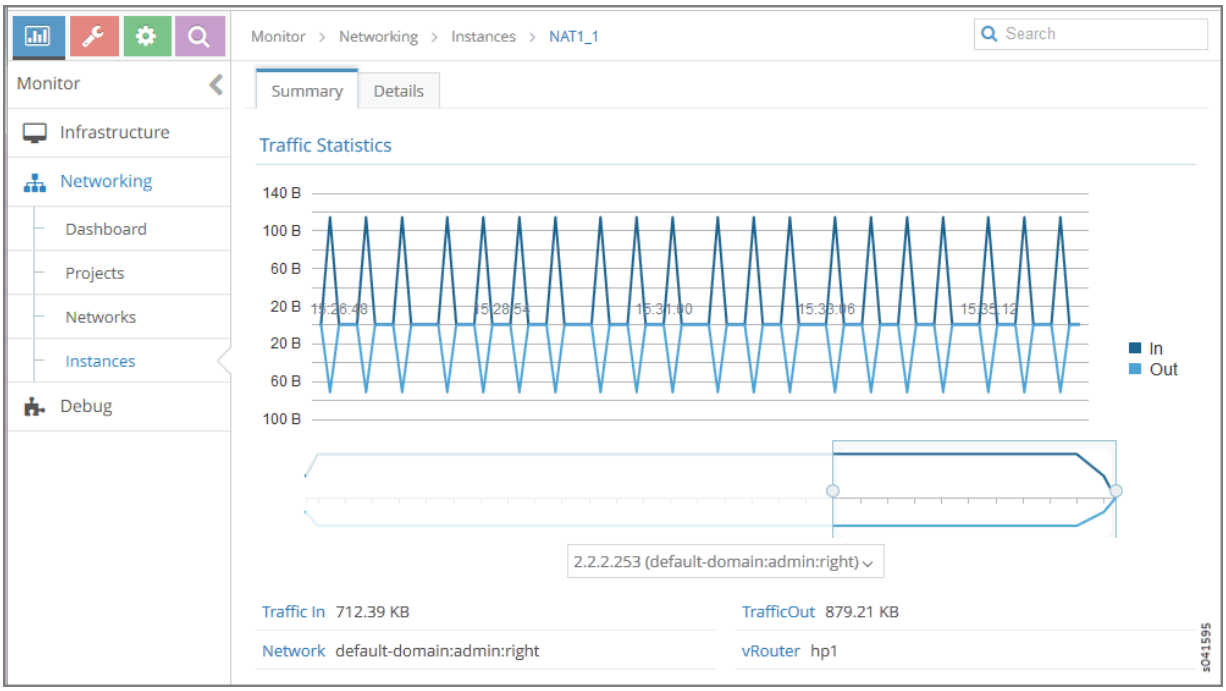
See Table 3 for a description of the fields on this screen.

Table 41: 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 63 on page 217 .

Figure 63: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See Figure 64 on page 217 .

Figure 64: Network Summary

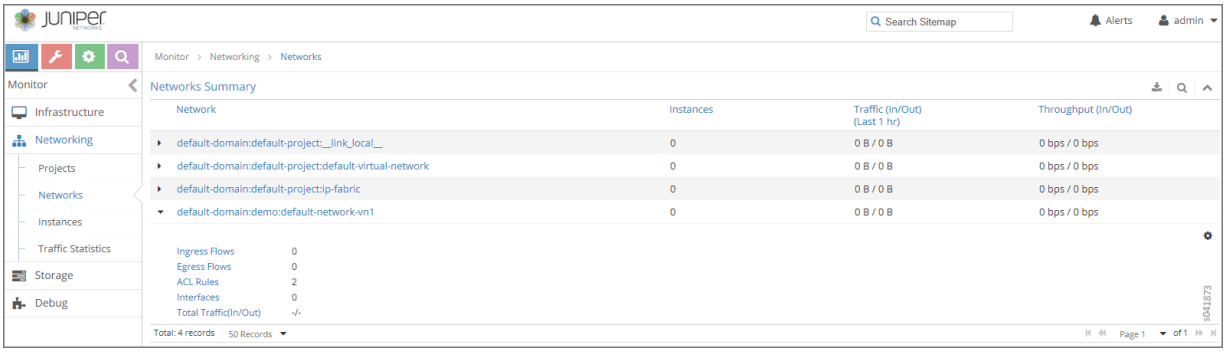


Table 42: Network Summary Fields

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 65 on page 218 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 65: Individual Network Connectivity Details—Summary Tab

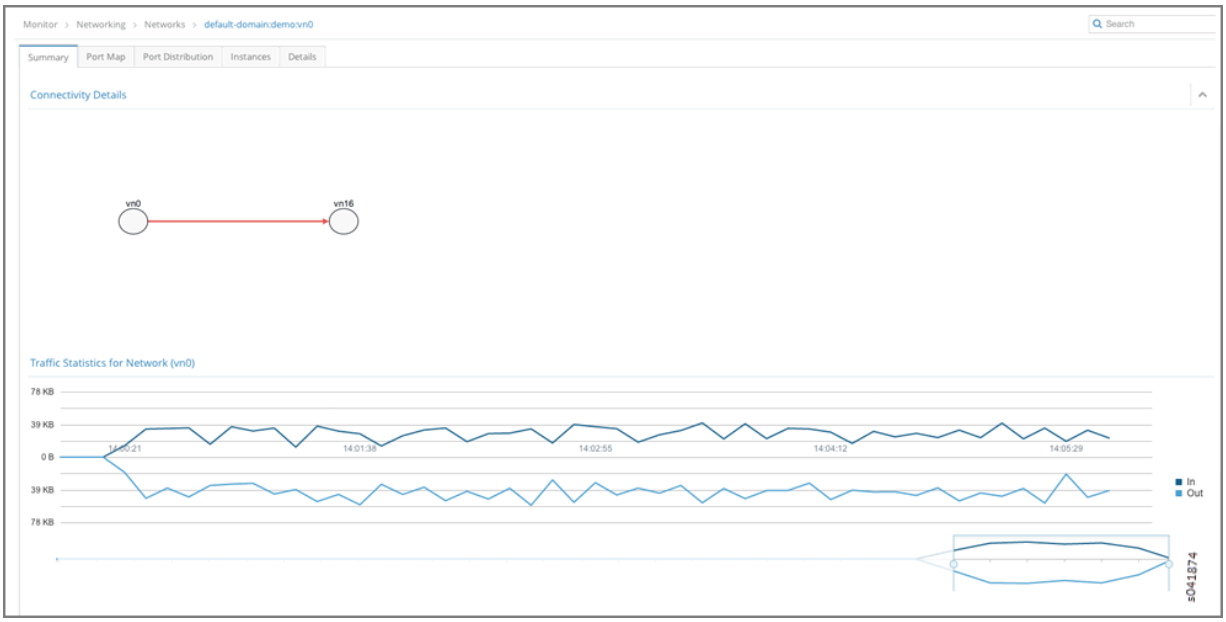


Figure 66 on page 219 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 66: Individual Network-– Port Map Tab

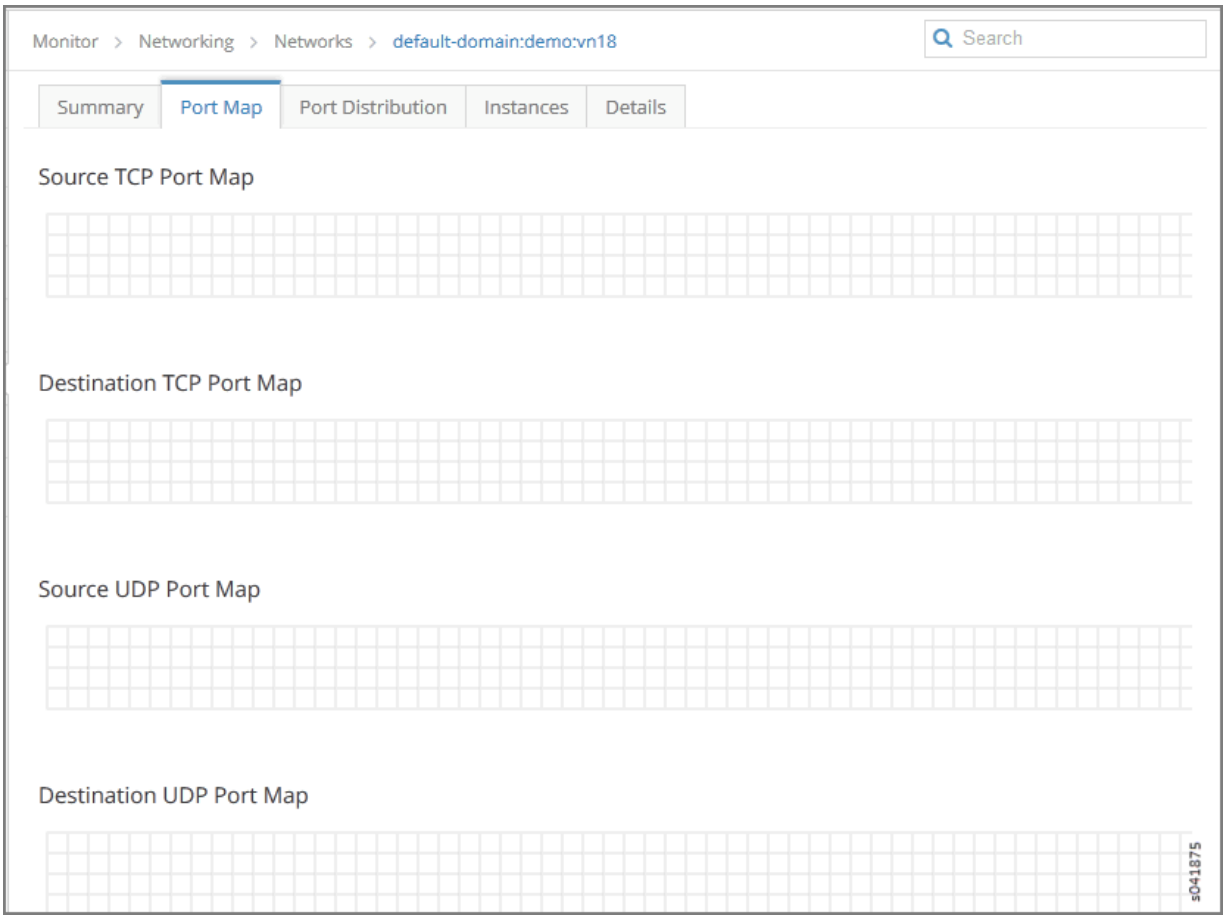


Figure 67 on page 220 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 67: Individual Network-– Port Distribution Tab

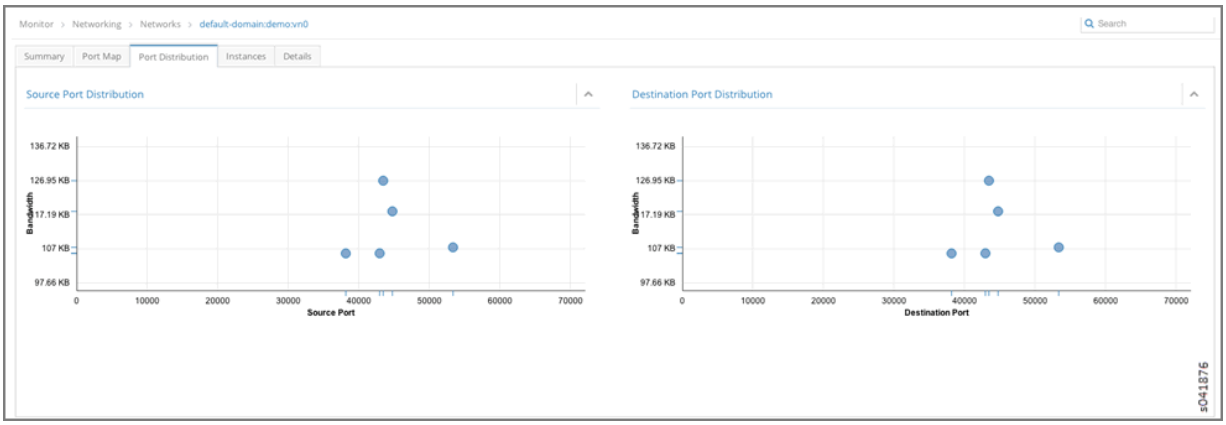


Figure 68 on page 221 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

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 68: Individual Network Instances Tab

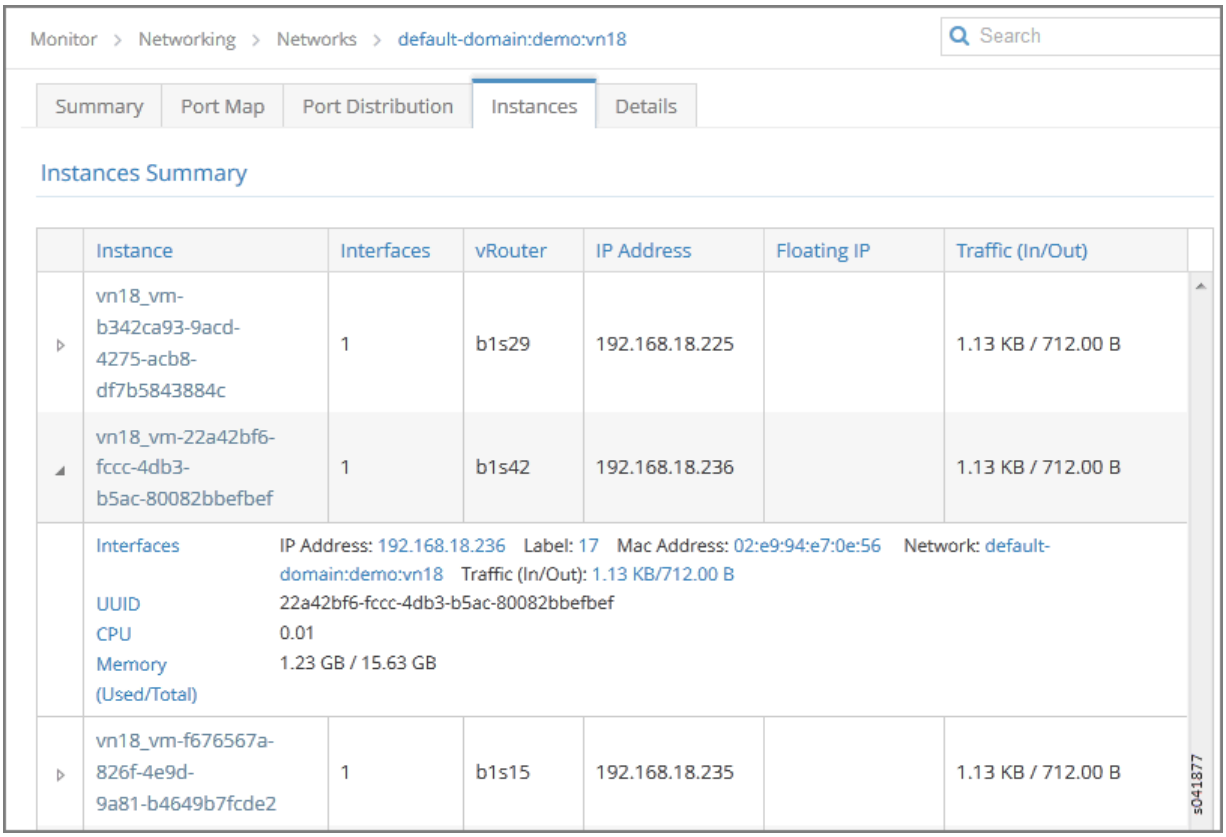
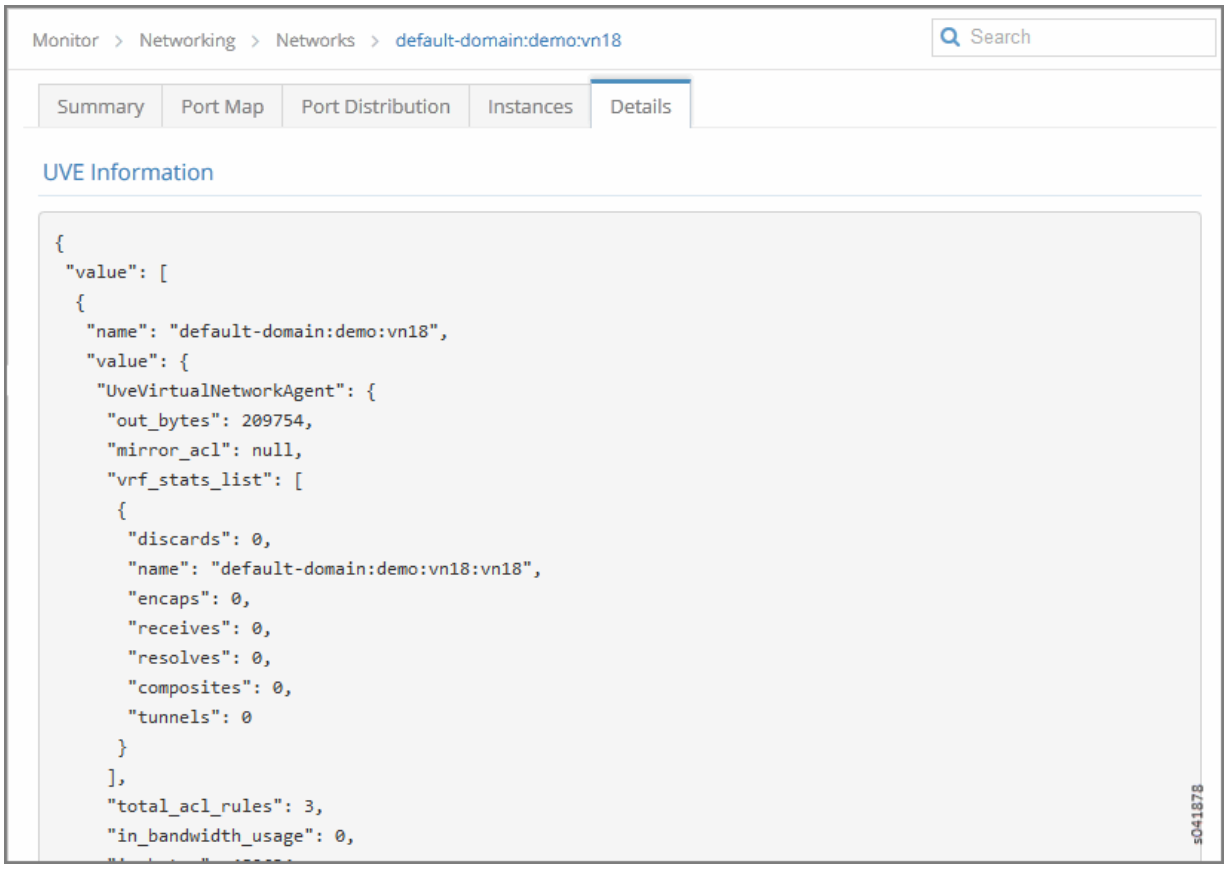


Figure 69 on page 222 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 69: Individual Network Details Tab



Query > Flows

IN THIS SECTION

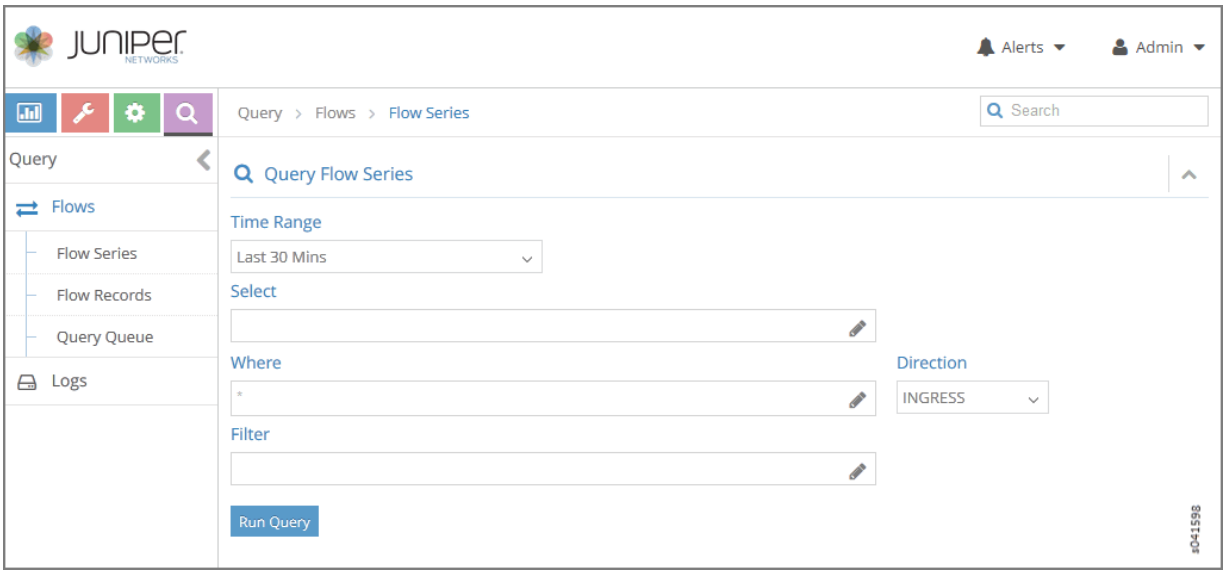
- Query > Flows > Flow Series | **223**
- Example: Query Flow Series | **226**
- Query > Flow Records | **228**
- Query > Flows > Query Queue | **231**

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 70 on page 223 .

Figure 70: Query Flow Series Window



The query fields available on the screen for the **Flow Series** tab are described in Table 43 on page 224 . Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 43: Query Flow Series Fields

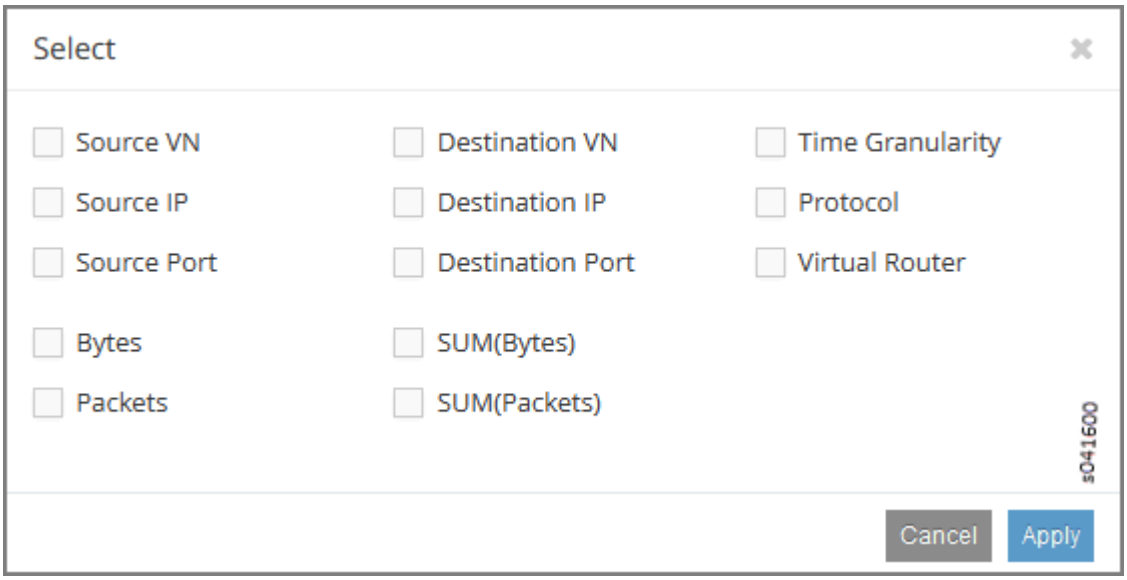
Field	Description
Time Range	Select a range of time to display the flow series: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specific custom time range in two fields: **From Time** and **To Time**.
Select	Click the edit button (pencil icon) to open a **Select** window (Figure 71 on page 225), 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 72 on page 226), 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.

Table 43: Query Flow Series Fields *(Continued)*

Field	Description
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.

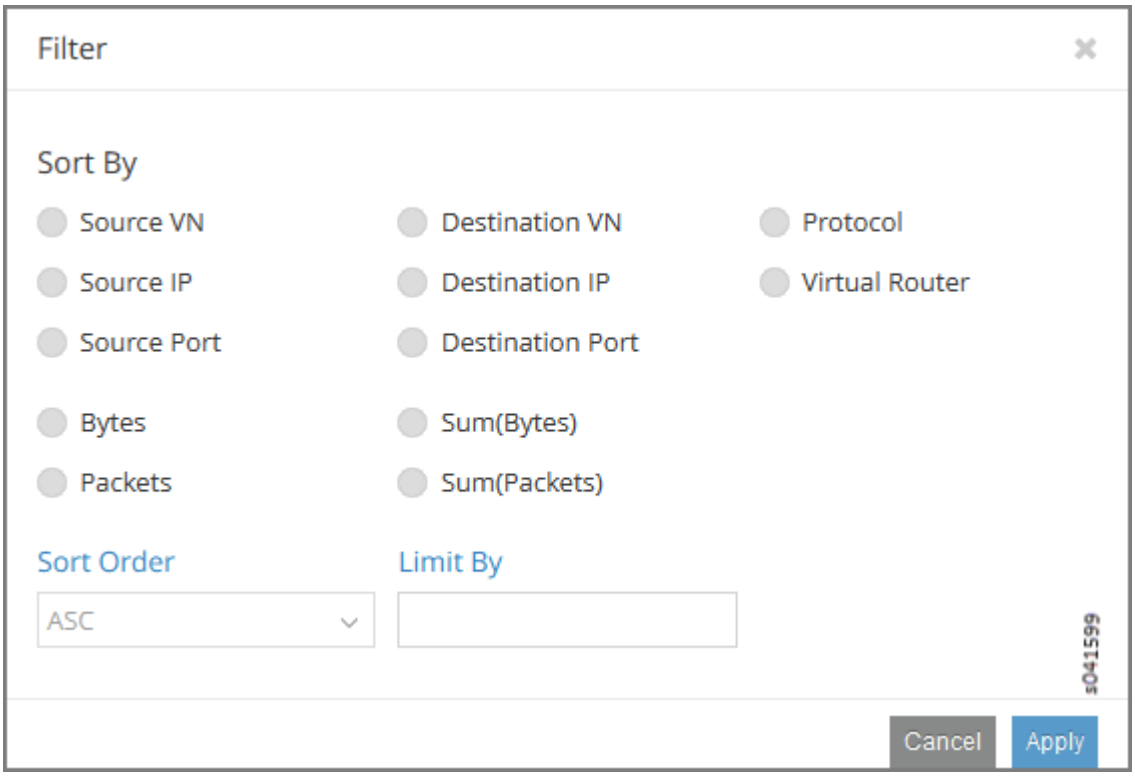
The **Select** window allows you to select one or more attributes of a flow series by clicking the check box for each attribute desired, see Figure 71 on page 225 . The upper section of the **Select** window includes field names, and the lower portion lets you select units. Select **Time Granularity** and then select **SUM(Bytes)** or **SUM(Packets)** to aggregate bytes and packets in intervals.

Figure 71: Flow Series Select



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 72 on page 226 .

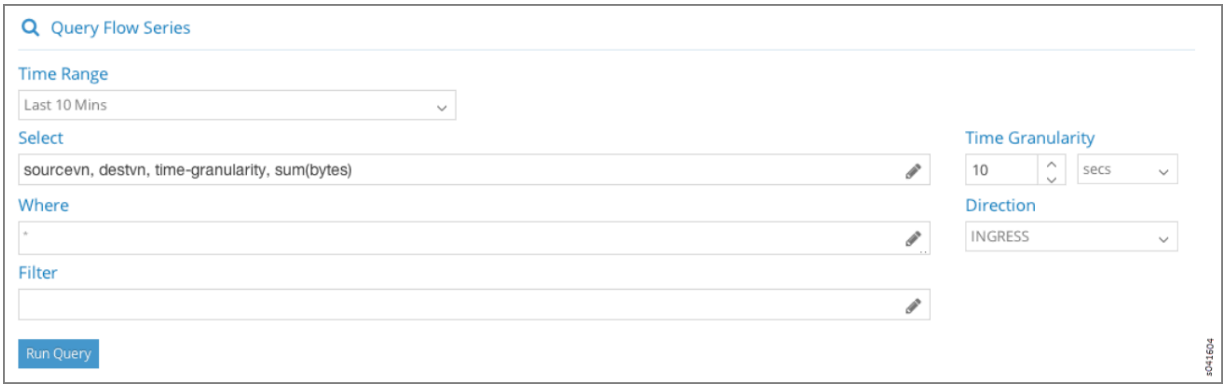
Figure 72: Flow Series Filter



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 73 on page 226 .

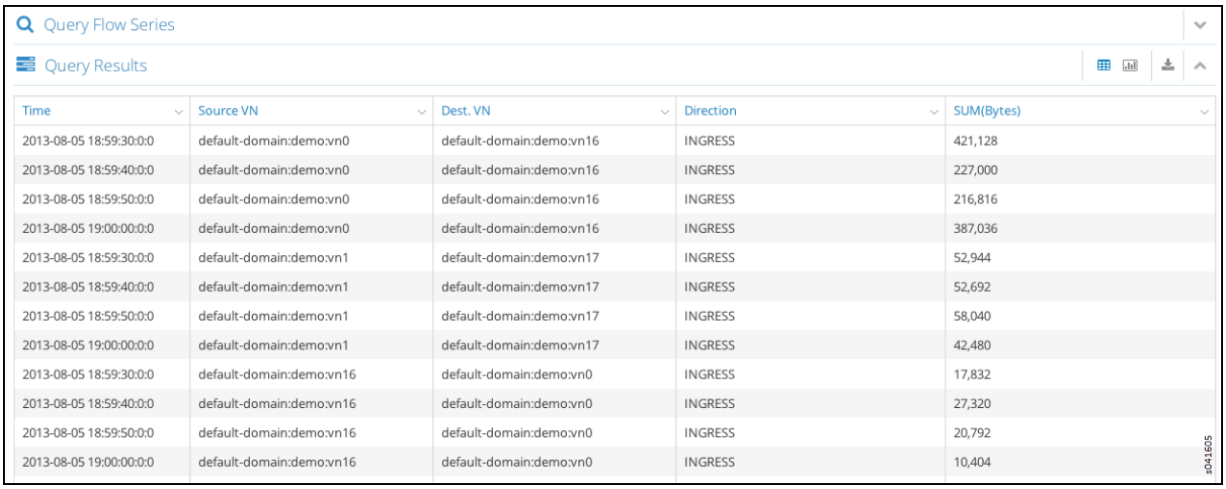
Figure 73: Example: Query Flow Series



The query returns tabular time series data, see Figure 74 on page 227 , for the following combinations of Source VN and Dest VN:

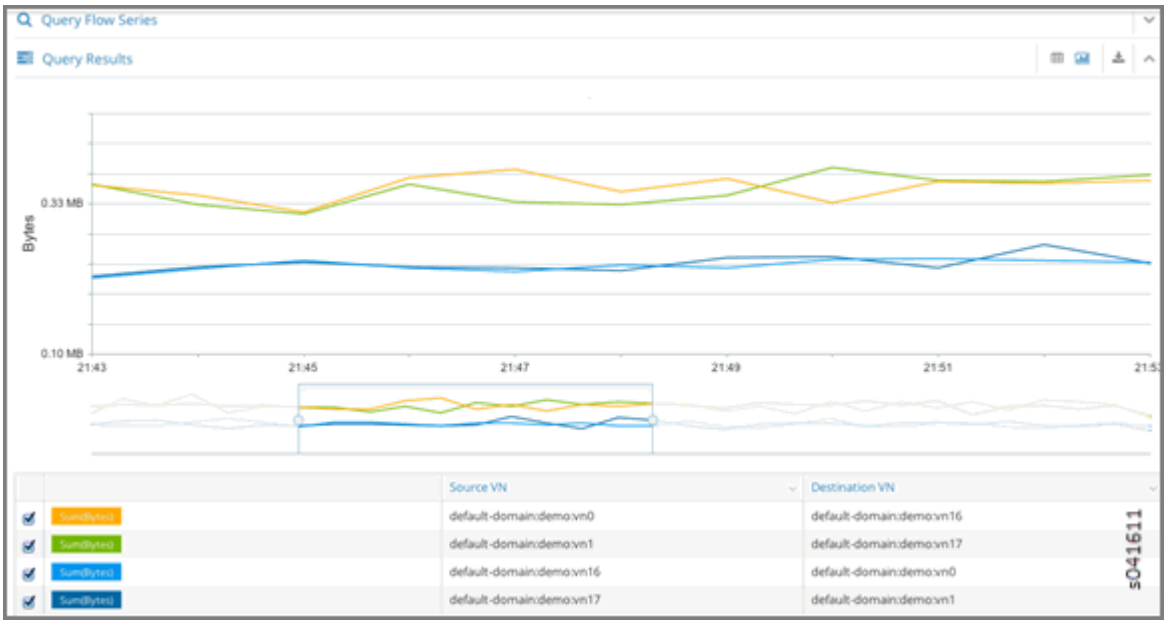
1. Flow Class 1: Source VN = default-domain:demo:front-end, Dest VN=__UNKNOWN__
2. Flow Class 2: Source VN = default-domain:demo:front-end, Dest VN=default-domain:demo:back-end

Figure 74: Query Flow Series Tabular Results



Because **Time Granularity** is selected, the results can also be displayed as graphical charts. Click the graph button on the right side of the tabular results. The results are displayed in a graphical flow chart. See Figure 75 on page 228 .

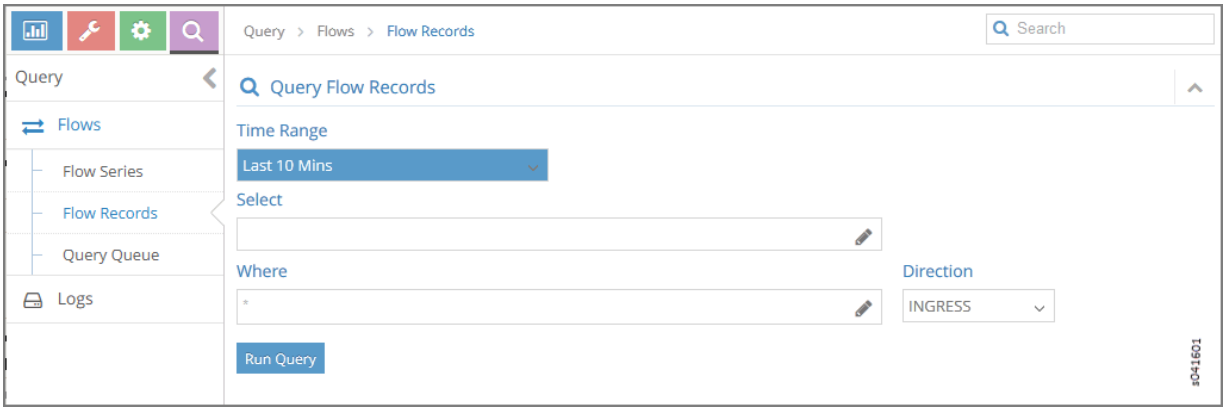
Figure 75: Query Flow Series Graphical Results



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 76: Flow Records



The query fields available on the screen for the **Flow Records** tab are described in Table 44 on page 229 . Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 44: Query Flow Records Fields

Field	Description
Time Range	Select a range of time for the flow records: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specified custom time range in two fields: **From Time** and **To Time**.
Select	Click the edit button (pencil icon) to open a **Select** window (Figure 77 on page 230), 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 77 on page 230 .

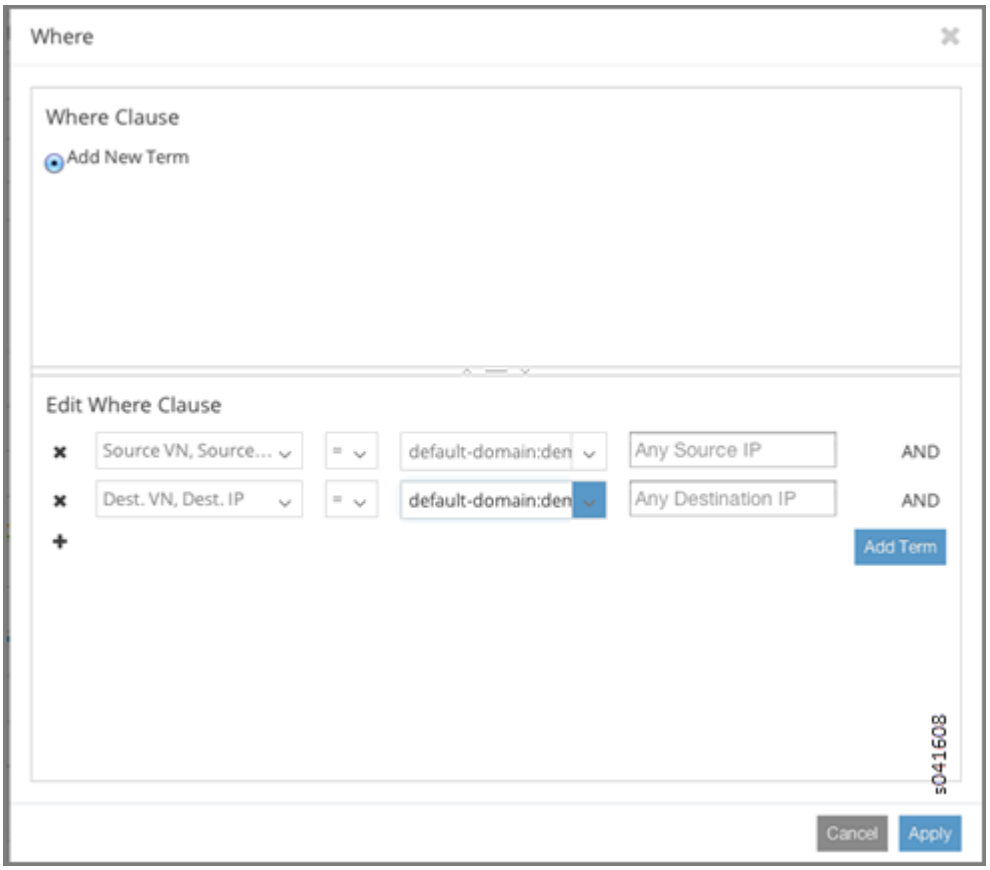
Figure 77: Flow Records Select Window

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**.

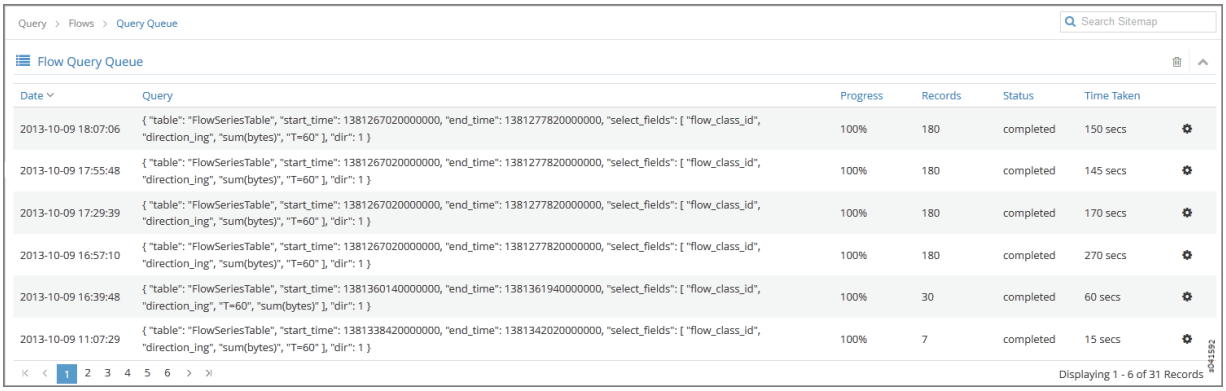
Figure 78: Where Clause Window



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 79 on page 231 .

Figure 79: Flows Query Queue



The query fields available on the screen for the **Flow Records** tab are described in Table 45 on page 232 . Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 45: 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

- Fat Flows

Query > Logs

IN THIS SECTION

- Query > Logs Menu Options | **233**
- Query > Logs > System Logs | **233**
- Sample Query for System Logs | **235**

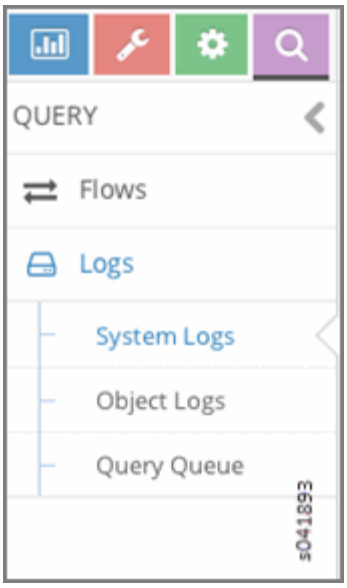
Query > Logs > Object Logs | 237

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 80 on page 233 .

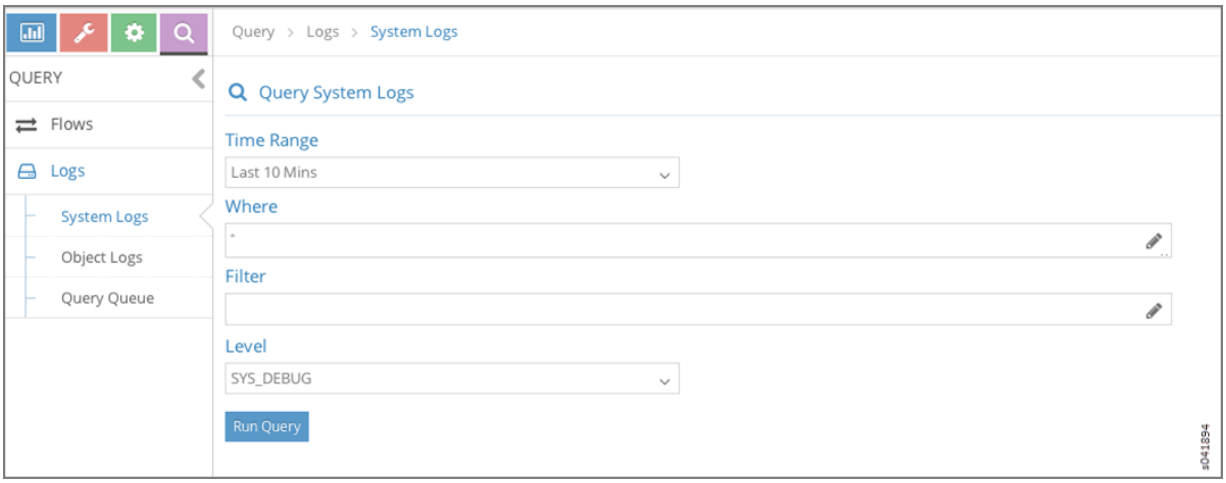
Figure 80: Query > Logs



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 81 on page 234 .

Figure 81: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in Table 46 on page 234 .

Table 46: Query System Logs Fields

Field	Description
Time Range	Select a range of time for which to see the system logs: • 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 46: Query System Logs Fields *(Continued)*

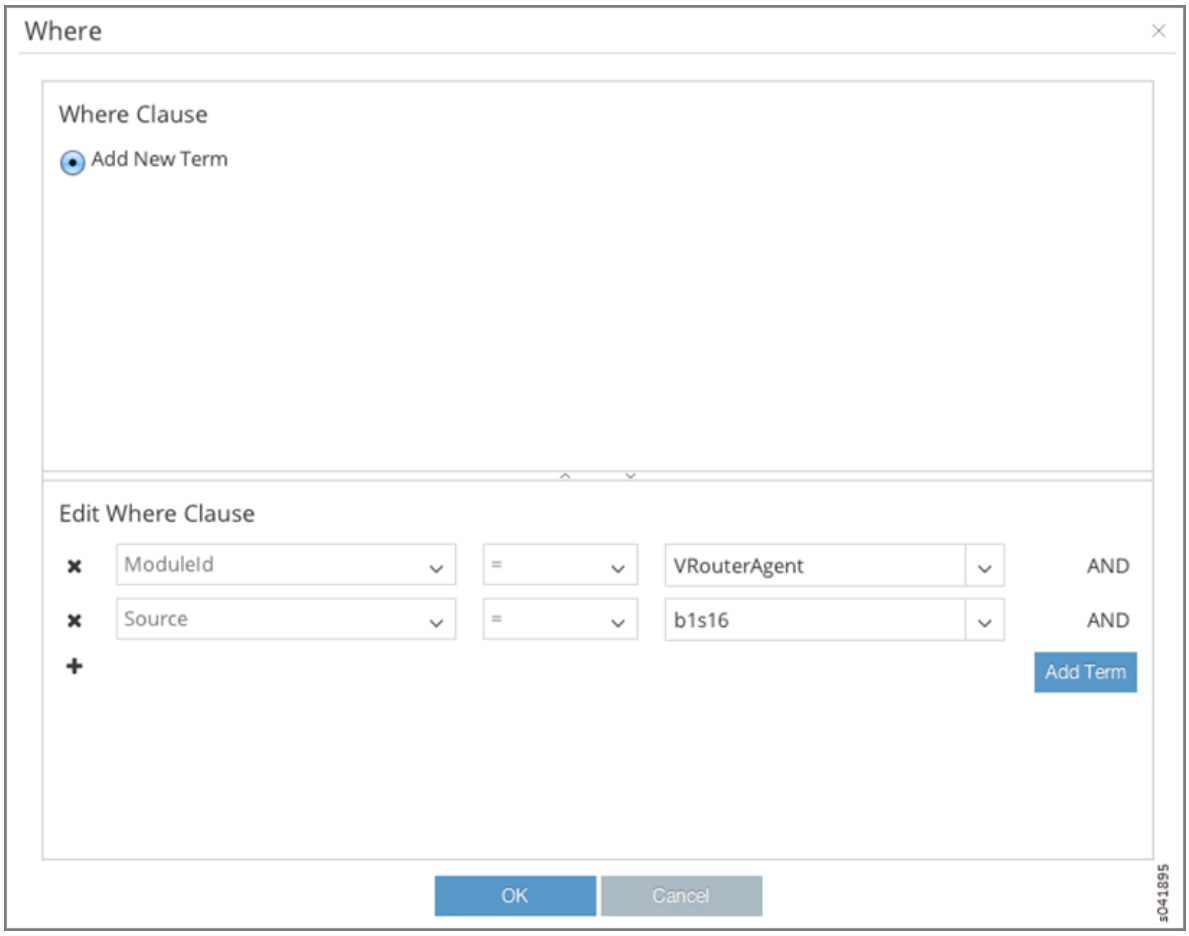
Field	Description
Level	Select the message severity level to view: • SYS_NOTICE • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_INFO • SYS_DEBUG
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the **Time**, **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 82 on page 236 .

Figure 82: Edit Where Clause



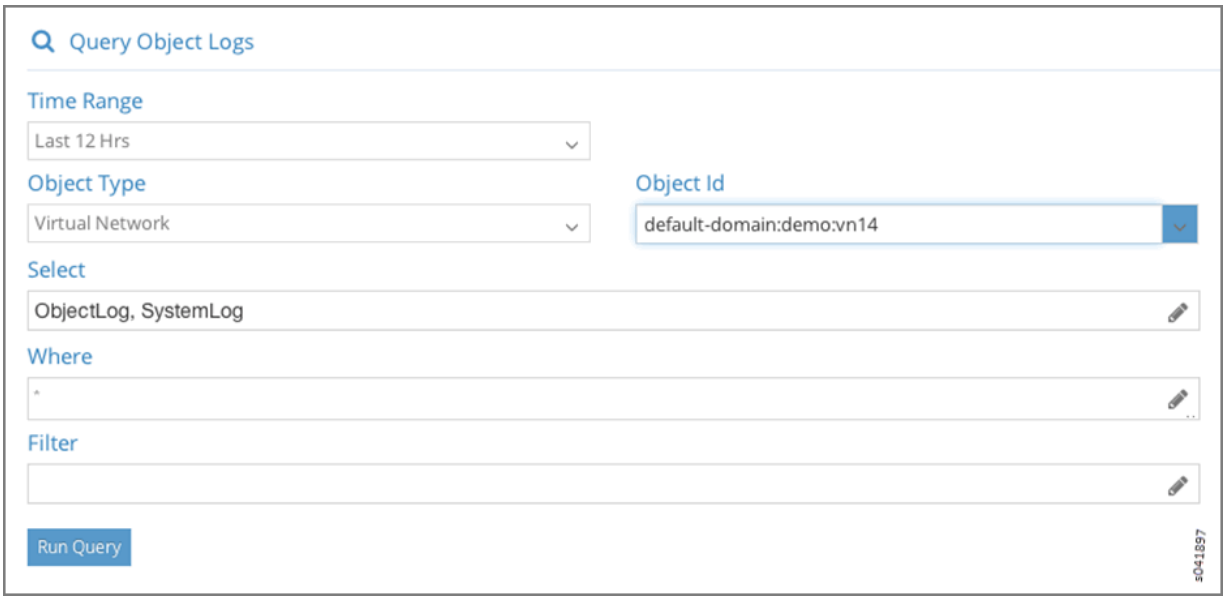
2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see Figure 83 on page 237 . When finished, click **Run Query** to display the results.

Figure 83: Sample Query System Logs

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 84 on page 237 .

Figure 84: Query > Logs > Object Logs



The query fields available on the **Object Logs** screen are described in Table 47 on page 238 .

Table 47: Object Logs Query Fields

Field	Description
Time Range	Select a range of time for which to see the logs: • 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**.
Object Type	Select the object type for which to show logs: • 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	Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox: • ObjectLog • SystemLog

Table 47: Object Logs Query Fields *(Continued)*

Field	Description
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.

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:

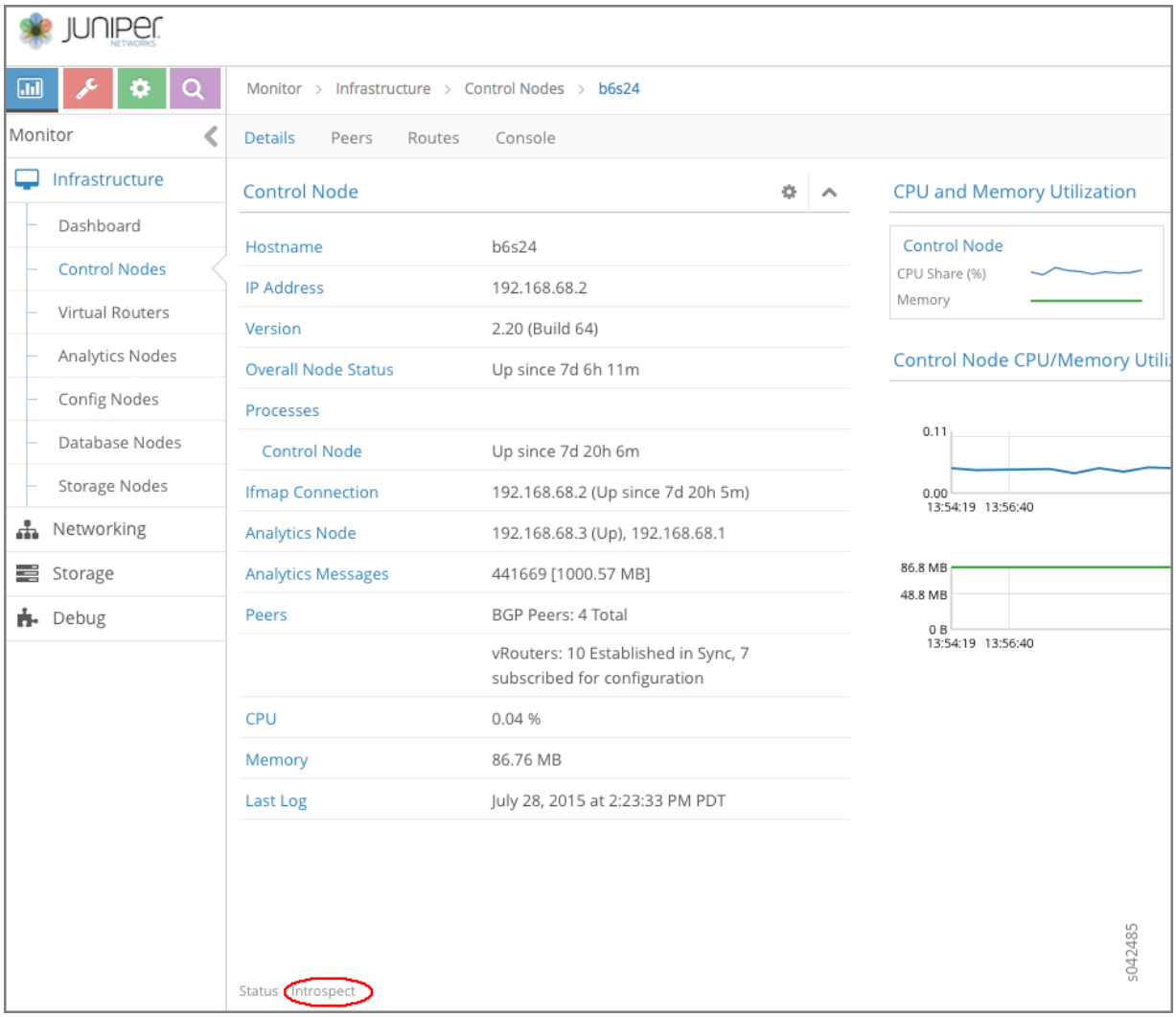
- *<controller-ip>*:8083. This port displays the *contrail-control* introspect port.
- *<compute-ip>*:8085 This port displays the *contrail-vrouter-agent* introspect port.
- *<controller-ip>*:8087 This port displays the *contrail-schema* introspect port.
- *<controller-ip>*:8088 This port displays the *contrail-svc-monitor* introspect port.
- *<controller-ip>*:8092 This port displays the *contrail-dns* introspect port.
- *<controller-ip>*:8084 This port displays the *contrail-api* introspect port. (:8084/Snh_SandeshTraceRequest?x=RestApiTraceBuf)

You can use the config editor to review configured objects.

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

Figure 85 on page 241 shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 85: 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 86 on page 242 shows the Controller Introspect window.

Figure 86: Controller Introspect Window

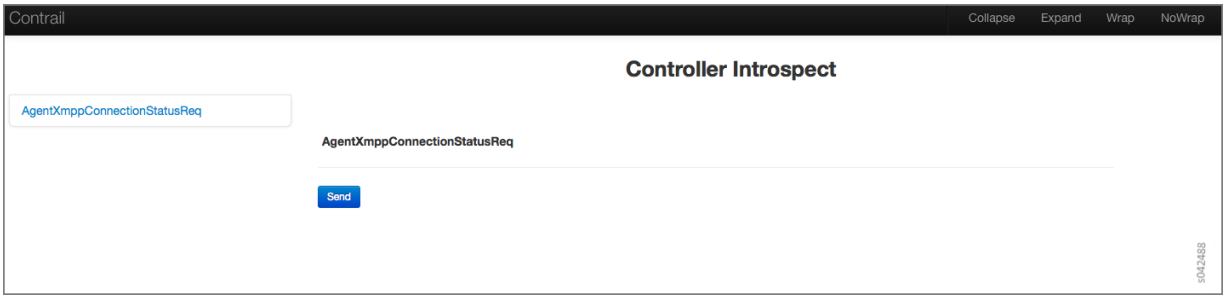


Figure 87 on page 242 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 87: BGP Peer Introspect Page

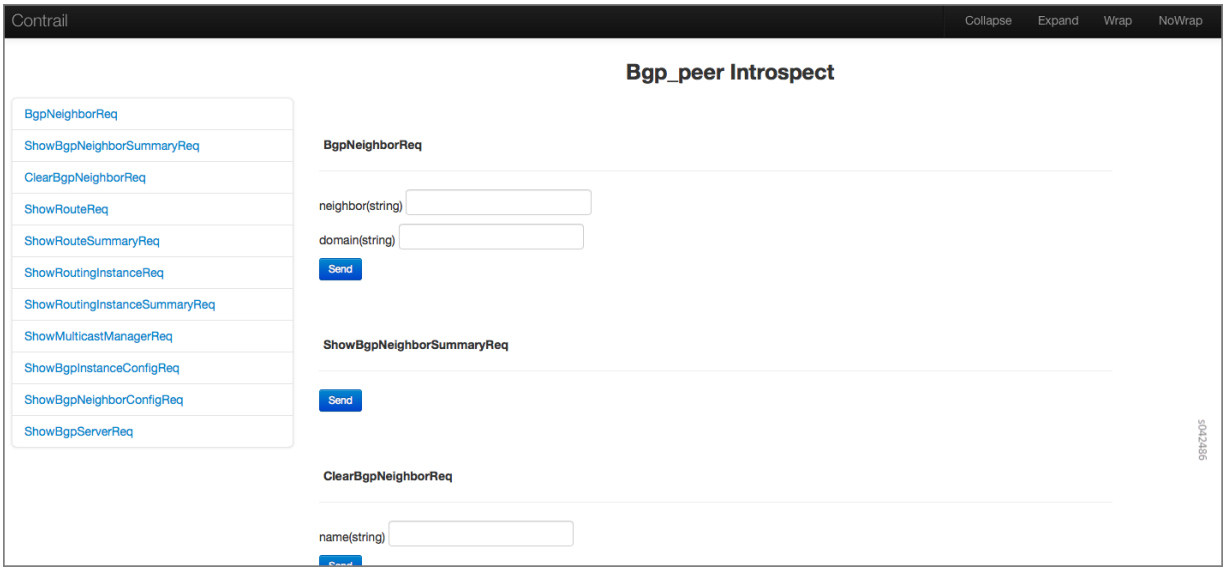
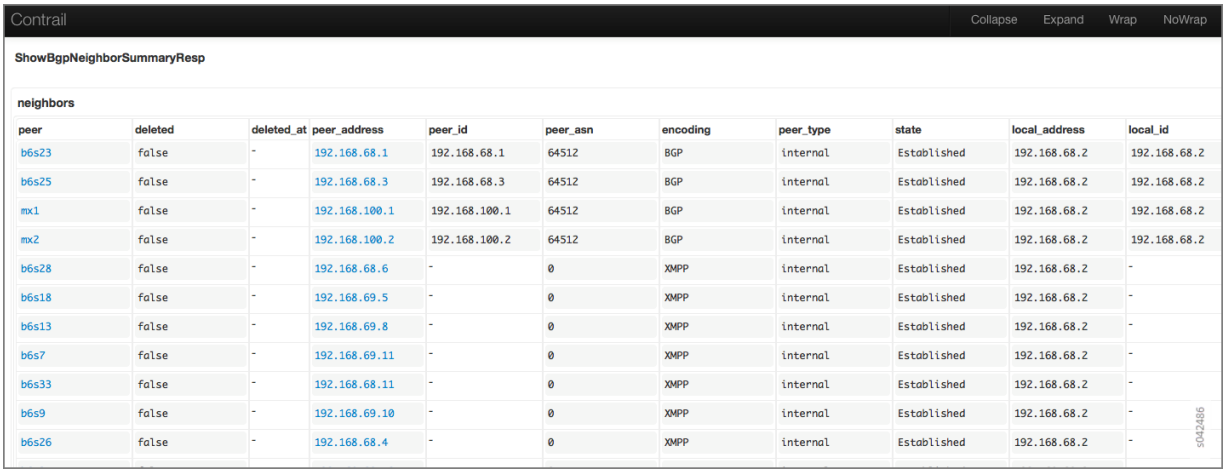


Figure 88 on page 243 shows an example of the BGP Neighbor Summary Introspect page.

Figure 88: BGP Neighbor Summary Introspect Page



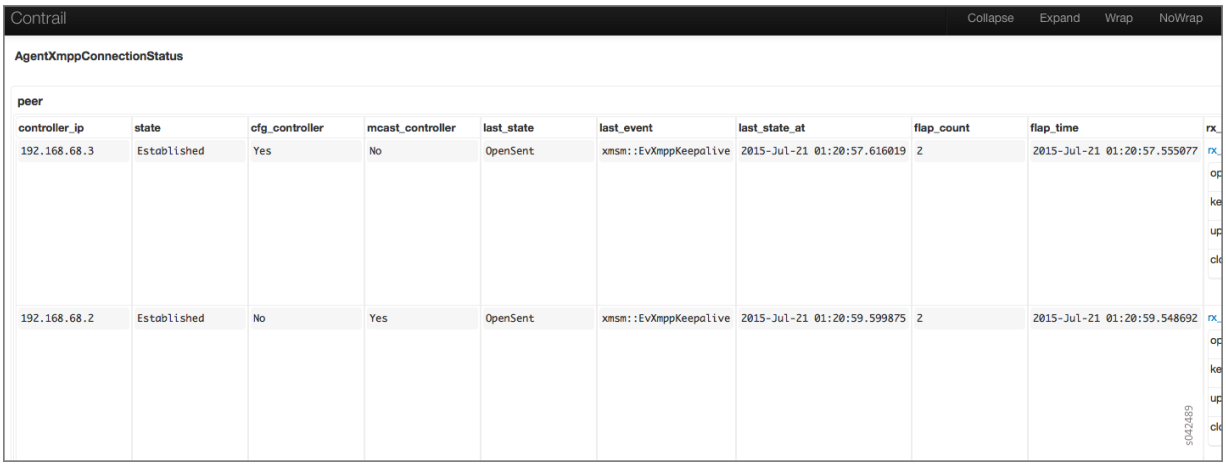
The following are the Sandesh modules for the Contrail vRouter agent (**contrail-vrouter-agent**) Introspect port.

- 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 89 on page 244 shows an example of the Agent (agent.xml) Introspect page.

Figure 89: Agent Introspect Page



RELATED DOCUMENTATION

Agent Modules in Contrail Networking | 130

Configuring Secure Sandesh and Introspect for Contrail Analytics | 142

Example: Debugging Connectivity Using Monitoring for Troubleshooting

IN THIS SECTION

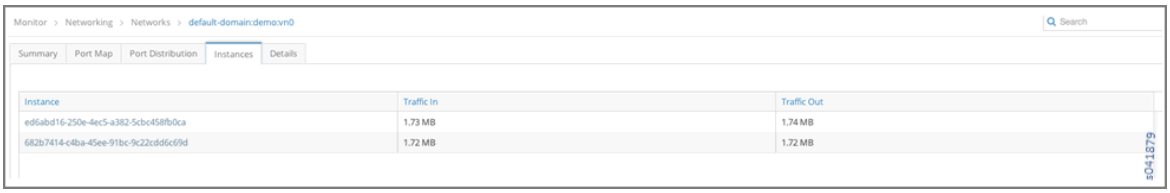
- Using Monitoring to Debug Connectivity | 245

Using Monitoring to Debug Connectivity

This example shows how you can use monitoring to debug connectivity in your Contrail system. You can use the demo setup in Contrail to use these steps on your own.

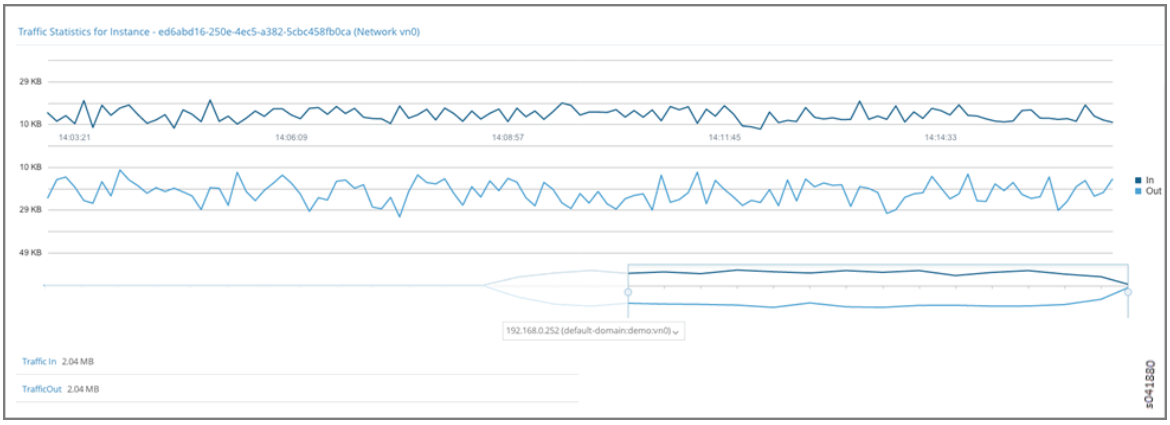
1. Navigate to **Monitor -> Networking -> Networks ->** `default-domain:demo:vn0`, **Instance** `ed6abd16-250e-4ec5-a382-5cbc458fb0ca` with **IP address** `192.168.0.252` in the virtual network `vn0`. See Figure 90 on page 245 .

Figure 90: Navigate to Instance



2. Click the instance to view **Traffic Statistics for Instance**. See Figure 91 on page 245 .

Figure 91: Traffic Statistics for Instance



3. **Instance** `d26c0b31-c795-400e-b8be-4d3e6de77dcf` with **IP address** `192.168.0.253` in the virtual network `vn16`. See Figure 92 on page 246 and Figure 93 on page 246 .

Figure 92: Navigate to Instance

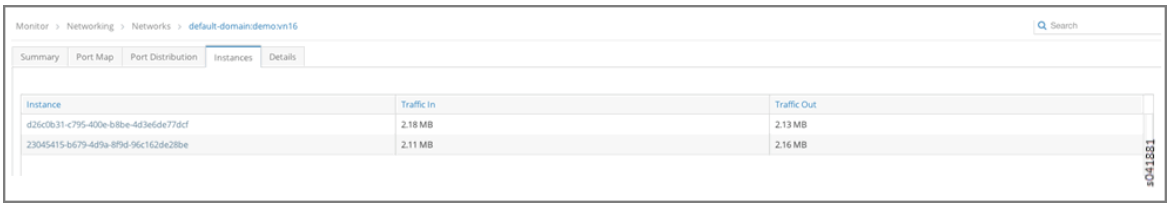
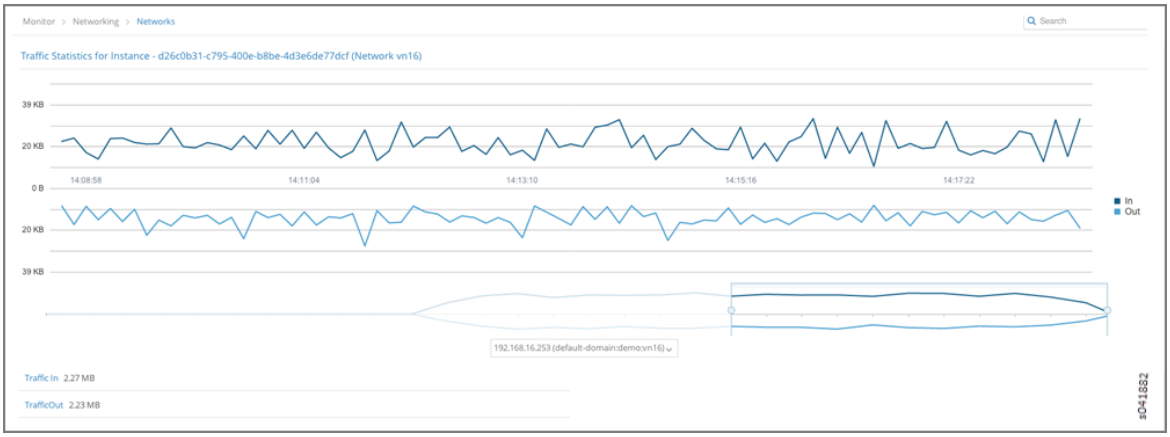
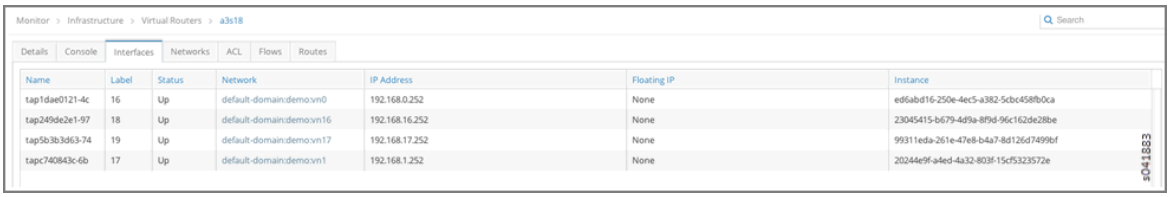


Figure 93: Traffic Statistics for Instance



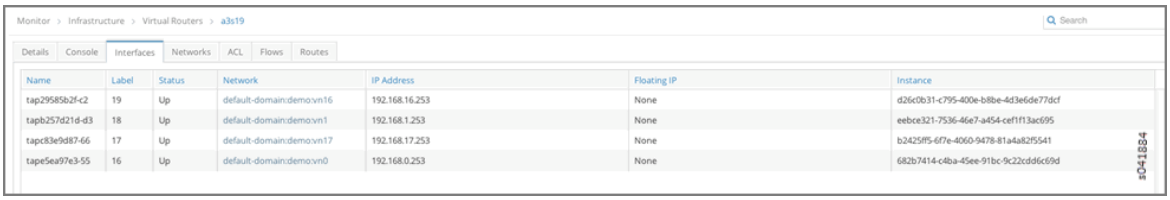
4. From **Monitor->Infrastructure->Virtual Routers->a3s18->Interfaces**, we can see that **Instance** `ed6abd16-250e-4ec5-a382-5cbc458fb0ca` is hosted on **Virtual Router** `a3s18`. See Figure 94 on page 246 .

Figure 94: Navigate to a3s18 Interfaces



5. From **Monitor->Infrastructure->Virtual Routers->a3s19->Interfaces**, we can see that **Instance** `d26c0b31-c795-400e-b8be-4d3e6de77dcf` is hosted on **Virtual Router** `a3s19`. See Figure 95 on page 247 .

Figure 95: Navigate to a3s19 Interfaces



6. **Virtual Routers** `a3s18` and `a3s19` have the **ACL** entries to allow connectivity between `default-domain:demo:vn0` and `default-domain:demo:vn16` networks. See Figure 96 on page 247 and Figure 97 on page 247 .

Figure 96: ACL Connectivity a3s18

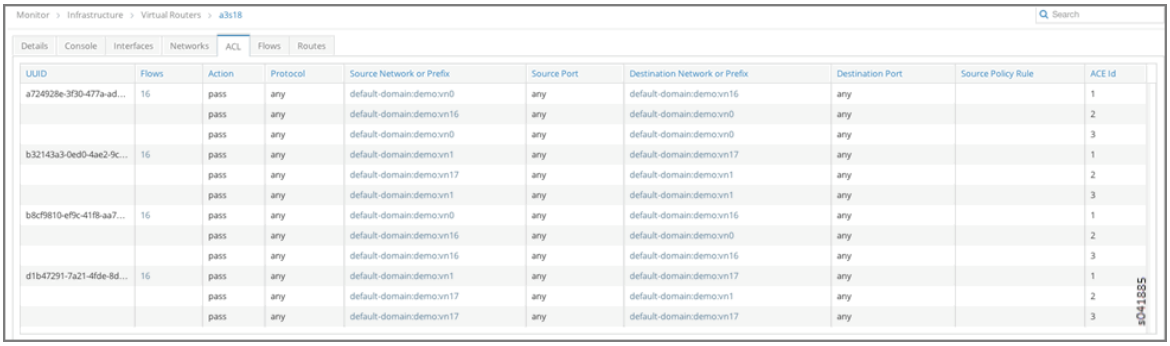
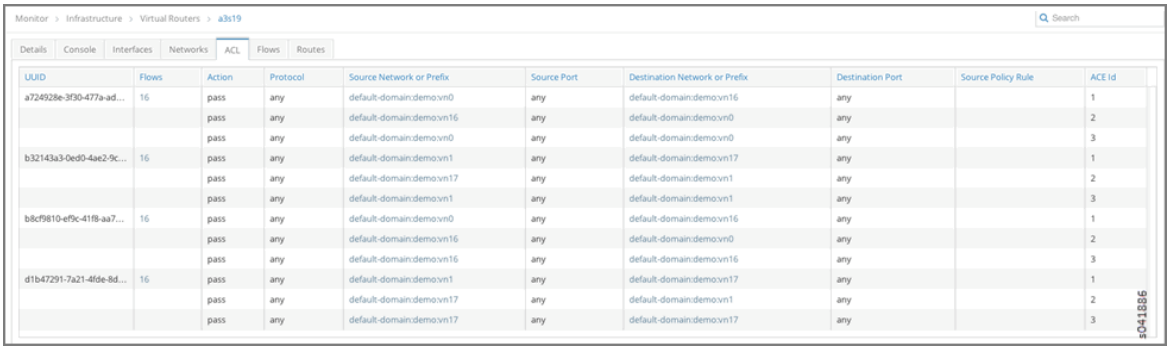


Figure 97: ACL Connectivity a3s19



7. Next, verify the routes on the control node for routing instances `default-domain:demo:vn0:vn0` and `default-domain:demo:vn16:vn16`. See Figure 98 on page 248 and Figure 99 on page 248 .

Figure 98: Routes default-domain:demo:vn0:vn0

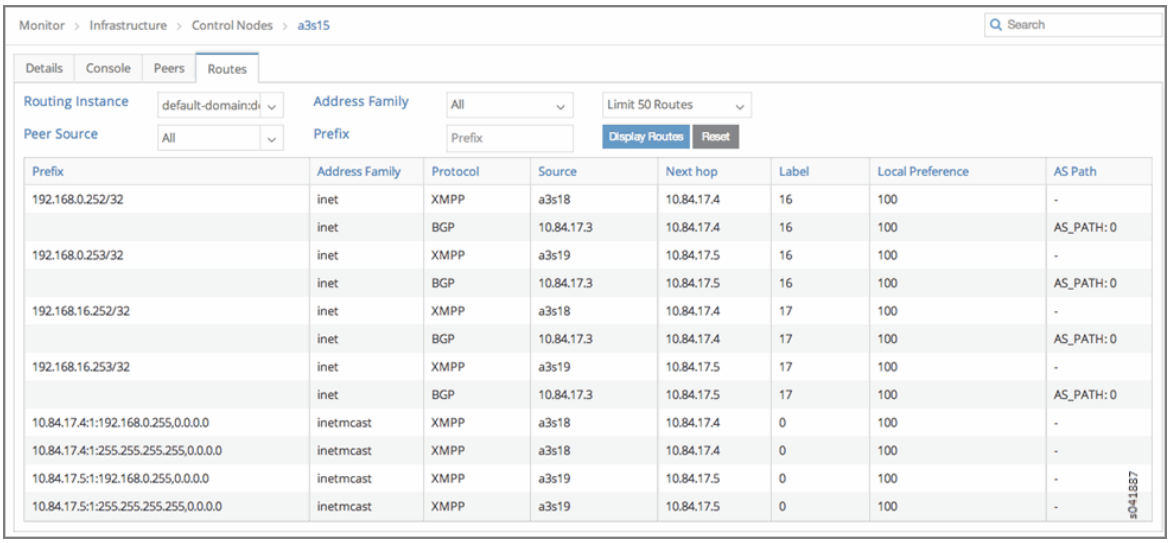
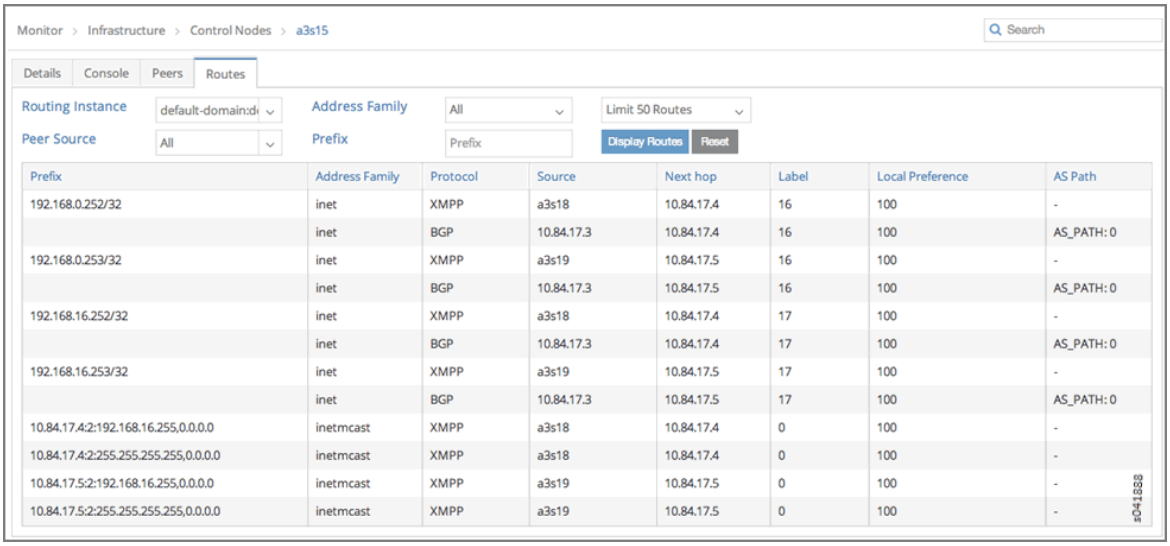
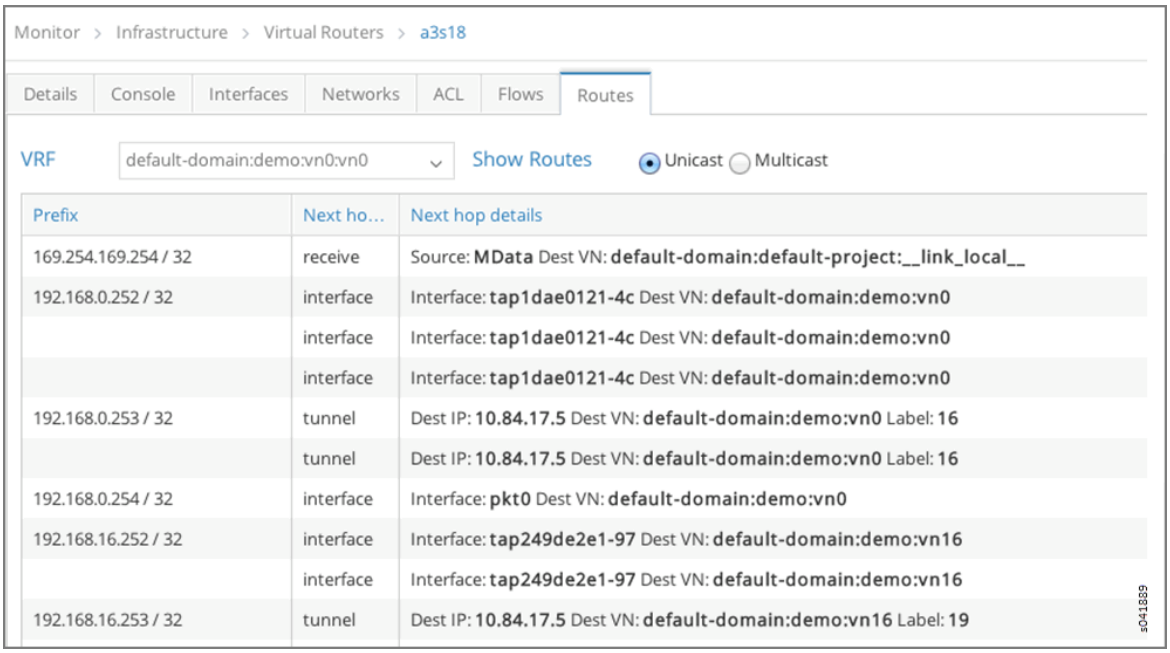


Figure 99: Routes default-domain:demo:vn16:vn16



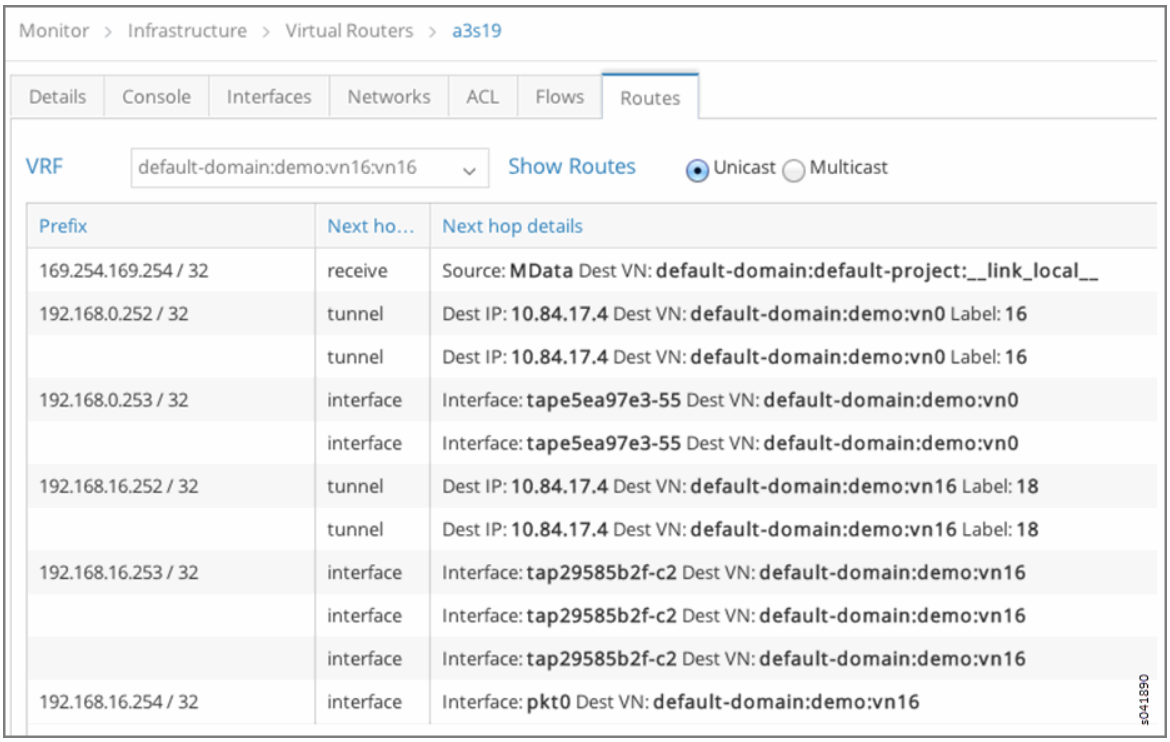
8. We can see that VRF `default-domain:demo:vn0:vn0` on Virtual Router `a3s18` has the appropriate route and next hop to reach VRF `default-domain:demo:front-end` on Virtual Router `a3s19`. See Figure 100 on page 249 .

Figure 100: Verify Route and Next Hop a3s18



9. We can see that VRF `default-domain:demo:vn16:vn16` on Virtual Router `a3s19` has the appropriate route and next hop to reach VRF `default-domain:demo:vn0:vn0` on Virtual Router `a3s18`. See Figure 101 on page 250 .

Figure 101: Verify Route and Next Hop a3s19



10. Finally, flows between instances (IPs `192.168.0.252` and `192.168.16.253`) can be verified on Virtual Routers `a3s18` and `a3s19`. See Figure 102 on page 250 and Figure 103 on page 251 .

Figure 102: Flows for a3s18

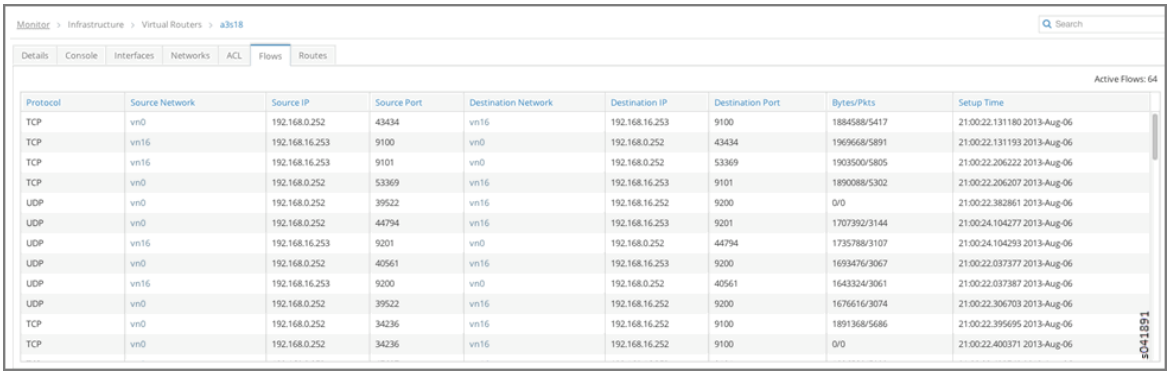
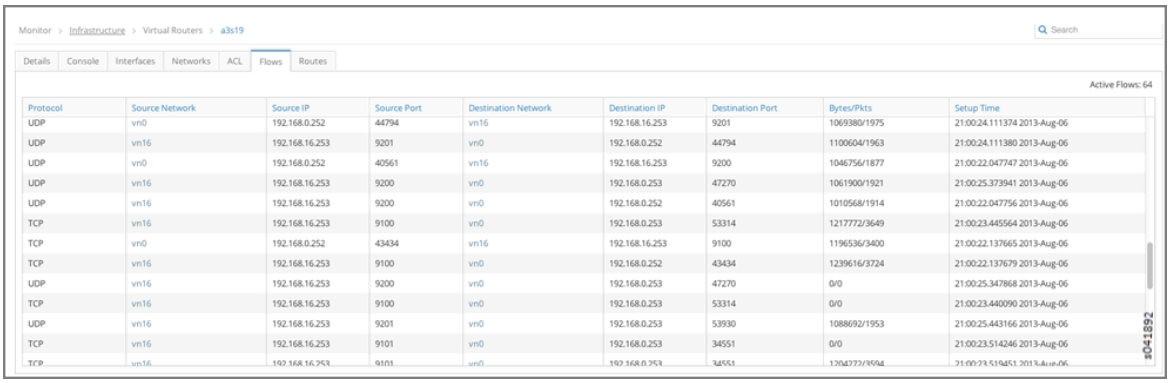


Figure 103: Flows for a3s19



Contrail Analytics Optional Modules

IN THIS SECTION

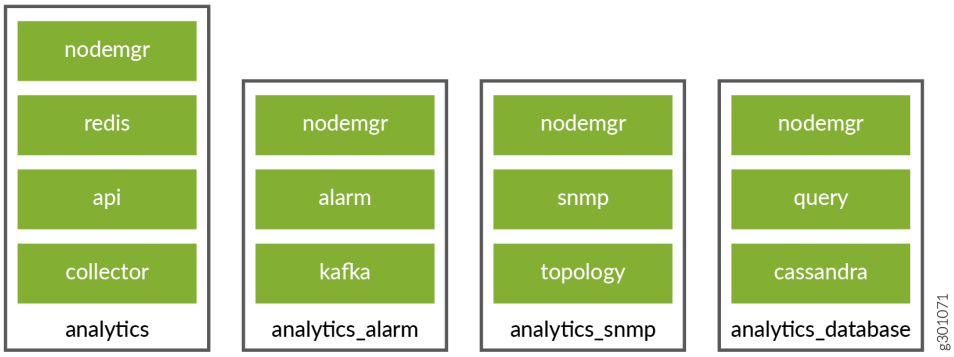
- Analytics Optional Components | **251**
- Contrail Web UI | **260**
- Tripleo Provisioning | **268**
- Appendix | **269**

Analytics Optional Components

Contrail analytics is comprised of four building blocks. The last three listed are optional components.

- Analytics collector
- Analytics alarm
- Analytics SNMP
- Analytics database

Figure 104: Contrail Analytics Components



Regardless that the alarm, SNMP, and database analytics roles have not been installed and if installed are disabled, these components show as active when you run the Linux $sudo commands or view in Contrail Command. For more information, see the section “TripleO Provisioning” below.

Contrail Infrastructure Installed without Optional Analytics Components

Two topologies are considered in this example: multi-nodes or single node.

Multi-nodes Contrail controller components are split onto three servers (Contrail controller, Contrail analytics, and Contrail analytics database). Only the first two servers are mandatory since Contrail analytics database is an optional component. This type of topology is used in production deployments.

Single node This type of topology is used in test deployments.

Multi-Nodes Contrail Controller

Contrail Controller Node

Following is an example of the Contrail status on the Contrail controller node:

```
$ sudo contrail-status
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active
```

```
== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active

== Contrail webui ==
web: active
job: active
```

Contrail Analytics Node (with All Optional Components)

Following is the Contrail status on Contrail analytics node when Contrail analytics SNMP and Contrail analytics alarm have both been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
analytics api contrail-analytics-api rhel-
queens-1910-23 running 62980f3e6479 Up 2 weeks
analytics collector contrail-analytics-collector rhel-
queens-1910-23 running b777437946c2 Up 2 weeks
analytics nodemgr contrail-nodemgr rhel-
queens-1910-23 running aeeb744a5b5e Up 2 weeks
analytics redis contrail-external-redis rhel-
queens-1910-23 running 150b6225bd93 Up 2 weeks
analytics-alarm alarm-gen contrail-analytics-alarm-gen rhel-
queens-1910-23 running d655146cb8d0 Up 2 weeks
analytics-alarm kafka contrail-external-kafka rhel-
queens-1910-23 running 8cfa8c7da4bd Up 2 weeks
analytics-alarm nodemgr contrail-nodemgr rhel-
queens-1910-23 running 685a5f817f0b Up 2 weeks
analytics-alarm zookeeper contrail-external-zookeeper rhel-
queens-1910-23 running a41dc5658c72 Up 2 weeks
```

```
analytics-snmp nodemgr contrail-nodemgr rhel-
queens-1910-23 running 0afd301ccbd8 Up 2 weeks
analytics-snmp snmp-collector contrail-analytics-snmp-collector rhel-
queens-1910-23 running 2bde6aa39250 Up 2 weeks
analytics-snmp topology contrail-analytics-snmp-topology rhel-
queens-1910-23 running a16f983ed162 Up 2 weeks

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active
```

Contrail analytics alarm and SNMP are deployed and active.

Contrail Analytics Node (without Analytics Optional Components)

Following is an example of the Contrail status on Contrail analytics node when Contrail analytics SNMP and Contrail analytics alarm have not been deployed:

```
$ sudo contrail-status
Pod Service Original Name Original Version
State Id Status
analytics api contrail-analytics-api rhel-queens-2005-62
running 489b07cbbbef Up 18 hours
analytics collector contrail-analytics-collector rhel-queens-2005-62
running 5da4f99b045f Up 18 hours
analytics nodemgr contrail-nodemgr rhel-queens-2005-62
running 28053f64f1bc Up 18 hours
analytics provisioner contrail-provisioner rhel-queens-2005-62
running faa8de6d17e4 Up 18 hours
analytics redis contrail-external-redis rhel-queens-2005-62
running 3e29dcc475d1 Up 18 hours
```

```
analytics stunnel contrail-external-stunnel rhel-queens-2005-62
running 11a30f0f5e3b Up 18 hours

== Contrail analytics ==
nodemgr: active
api: active
collector: active
```

Only Contrail analytics collector is deployed and active.

Contrail Analytics Database Node

Contrail analytics database is only deployed when the analytics database component is enabled. The following example shows the Contrail status on the Contrail analytics database node:

```
$ sudo contrail-status
Pod Service Original Name Original Version
State Id Status
database cassandra contrail-external-cassandra rhel-queens-1910-
23 running ec05bd8c34c4 Up 2 weeks
database nodemgr contrail-nodemgr rhel-queens-1910-
23 running 25a6c58d5144 Up 2 weeks
database query-engine contrail-analytics-query-engine rhel-queens-1910-
23 running f90f7ae16b48 Up 2 weeks

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active
```

Single Node Contrail Controller

Contrail Controller Node (with All Analytics Optional Components)

Following is the Contrail status on Contrail controller node when Contrail analytics SNMP, Contrail analytics alarm, and Contrail analytics database have been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
```

```
analytics api contrail-analytics-api
rhel-queens-1912-46 running bf87cc51fb36 Up 8 weeks
analytics collector contrail-analytics-collector
rhel-queens-1912-46 running 0ae1ca0fb1f2 Up 8 weeks
analytics nodemgr contrail-nodemgr
rhel-queens-1912-46 running 24e9174056d0 Up 8 weeks
analytics redis contrail-external-redis
rhel-queens-1912-46 running 9d7135b6b9d8 Up 8 weeks
analytics stunnel contrail-external-stunnel
rhel-queens-1912-46 running 30d413bad4f1 Up 8 weeks
analytics-alarm alarm-gen contrail-analytics-alarm-gen
rhel-queens-1912-46 running 2f40aeb42154 Up 8 weeks
analytics-alarm kafka contrail-external-kafka
rhel-queens-1912-46 running 8cd54b9520af Up 8 weeks
analytics-alarm nodemgr contrail-nodemgr
rhel-queens-1912-46 running afeadd231273 Up 8 weeks
analytics-alarm zookeeper contrail-external-zookeeper
rhel-queens-1912-46 running 118b116b2721 Up 8 weeks
analytics-snmp nodemgr contrail-nodemgr
rhel-queens-1912-46 running f623346fff53 Up 8 weeks
analytics-snmp snmp-collector contrail-analytics-snmp-collector
rhel-queens-1912-46 running 152b037af72d Up 8 weeks
analytics-snmp topology contrail-analytics-snmp-topology
rhel-queens-1912-46 running 5226847e74f3 Up 8 weeks
config api contrail-controller-config-api
rhel-queens-1912-46 running b8ba22697cfe Up 8 weeks
config device-manager contrail-controller-config-devicemgr
rhel-queens-1912-46 running 29f9b248f850 Up 8 weeks
config nodemgr contrail-nodemgr
rhel-queens-1912-46 running 2f3f84d5d2b4 Up 8 weeks
config schema contrail-controller-config-schema
rhel-queens-1912-46 running 334906b962fb Up 8 weeks
config svc-monitor contrail-controller-config-svcmonitor
rhel-queens-1912-46 running a8581c37f9ab Up 8 weeks
config-database cassandra contrail-external-cassandra
rhel-queens-1912-46 running e47a3e430fe6 Up 8 weeks
config-database nodemgr contrail-nodemgr
rhel-queens-1912-46 running 4798399f0ec5 Up 8 weeks
config-database rabbitmq contrail-external-rabbitmq
rhel-queens-1912-46 running d80a5e8e8801 Up 8 weeks
config-database zookeeper contrail-external-zookeeper
rhel-queens-1912-46 running b1c430201497 Up 8 weeks
control control contrail-controller-control-control
```

```
rhel-queens-1912-46 running e478128385f7 Up 8 weeks
control dns contrail-controller-control-dns
rhel-queens-1912-46 running f9752a324d71 Up 8 weeks
control named contrail-controller-control-named
rhel-queens-1912-46 running 66c992adced5 Up 8 weeks
control nodemgr contrail-nodemgr
rhel-queens-1912-46 running 3c9a0270ab1a Up 8 weeks
database cassandra contrail-external-cassandra
rhel-queens-1912-46 running f85ead18fb26 Up 8 weeks
database nodemgr contrail-nodemgr
rhel-queens-1912-46 running 0d9f471003ea Up 8 weeks
database query-engine contrail-analytics-query-engine
rhel-queens-1912-46 running 40a092abbccf Up 8 weeks
webui job contrail-controller-webui-job
rhel-queens-1912-46 running 432f686a8abf Up 8 weeks
webui web contrail-controller-webui-web
rhel-queens-1912-46 running 4341432ce9a4 Up 8 weeks

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
```

```
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

Contrail database (query), analytics alarm, and SNMP are deployed and active.

Contrail Controller Node (without Analytics Optional Components)

Following is an example of the Contrail status on Contrail controller node when Contrail analytics SNMP, Contrail analytics alarm, and Contrail analytics database have not been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
analytics api contrail-analytics-api
rhel-queens-2005-62 running b1ddca562595 Up 10 hours
analytics collector contrail-analytics-collector
rhel-queens-2005-62 running f6860911ee16 Up 10 hours
analytics nodemgr contrail-nodemgr
rhel-queens-2005-62 running 37a0d8744e31 Up 10 hours
analytics provisioner contrail-provisioner
rhel-queens-2005-62 running e2f9a4605d63 Up 10 hours
analytics redis contrail-external-redis
rhel-queens-2005-62 running 1d0a193983b0 Up 10 hours
analytics stunnel contrail-external-stunnel
rhel-queens-2005-62 running 695d61045e63 Up 10 hours
config api contrail-controller-config-api
rhel-queens-2005-62 running 41eb0caef12d Up 10 hours
```

```
config device-manager contrail-controller-config-devicemgr
rhel-queens-2005-62 running f3158c67d792 Up 10 hours
config nodemgr contrail-nodemgr
rhel-queens-2005-62 running 4138cc386e69 Up 10 hours
config provisioner contrail-provisioner
rhel-queens-2005-62 running 45aae86bb41a Up 10 hours
config schema contrail-controller-config-schema
rhel-queens-2005-62 running 2497392980d0 Up 10 hours
config svc-monitor contrail-controller-config-svcmonitor
rhel-queens-2005-62 running b2ed20209aa7 Up 10 hours
config-database cassandra contrail-external-cassandra
rhel-queens-2005-62 running abd3efad8075 Up 10 hours
config-database nodemgr contrail-nodemgr
rhel-queens-2005-62 running bcc74ecb37cc Up 10 hours
config-database provisioner contrail-provisioner
rhel-queens-2005-62 running 9de114119be5 Up 10 hours
config-database rabbitmq contrail-external-rabbitmq
rhel-queens-2005-62 running d623f5d3da79 Up 10 hours
config-database zookeeper contrail-external-zookeeper
rhel-queens-2005-62 running 2c4f47c2fdc1 Up 10 hours
control control contrail-controller-control-control
rhel-queens-2005-62 running 56e238791c60 Up 10 hours
control dns contrail-controller-control-dns
rhel-queens-2005-62 running 6cfc801451f9 Up 10 hours
control named contrail-controller-control-named
rhel-queens-2005-62 running f033a8bf5b88 Up 10 hours
control nodemgr contrail-nodemgr
rhel-queens-2005-62 running 7381053ff80f Up 10 hours
control provisioner contrail-provisioner
rhel-queens-2005-62 running a3851c25f427 Up 10 hours
webui job contrail-controller-webui-job
rhel-queens-2005-62 running 80cd5c06ff39 Up 10 hours
webui web contrail-controller-webui-web
rhel-queens-2005-62 running 51a2f164a259 Up 10 hours

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics ==
nodemgr: active
```

```
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active

== Contrail webui ==
web: active
job: active
```

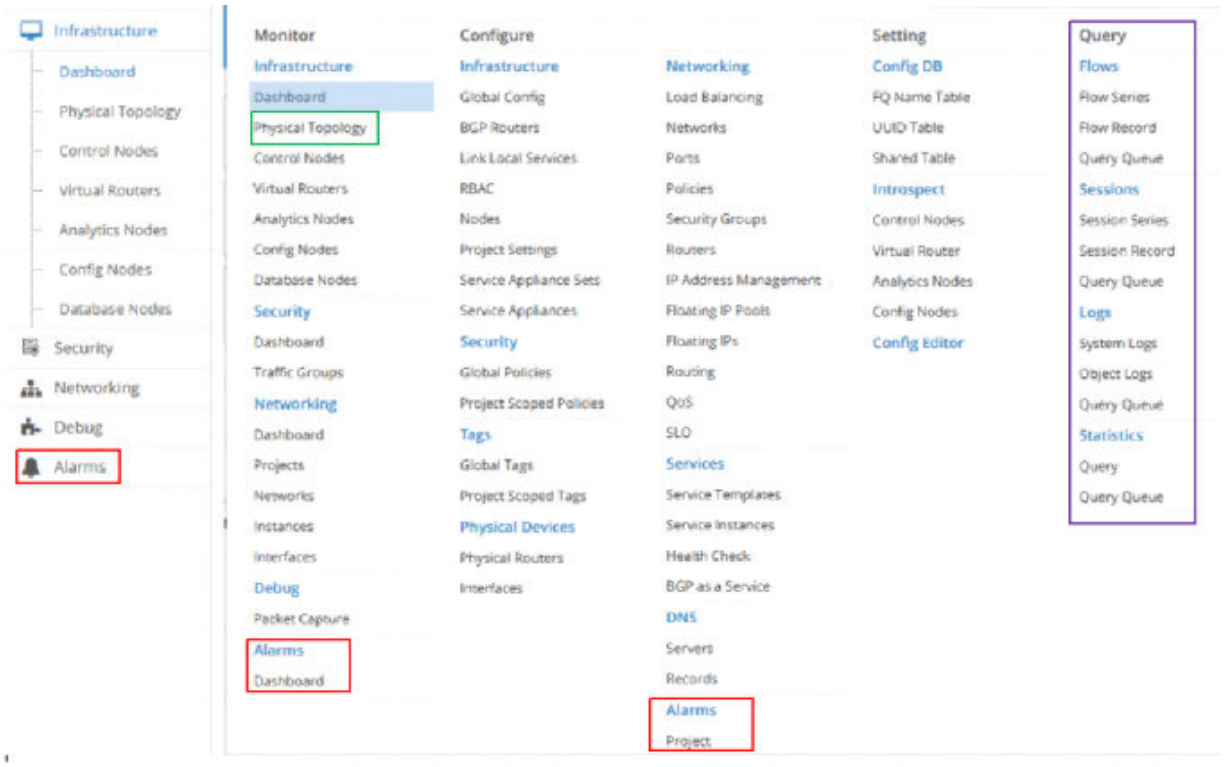
Contrail database (query), analytics alarm, and SNMP are not deployed.

Contrail Web UI

Web UI with Optional Components

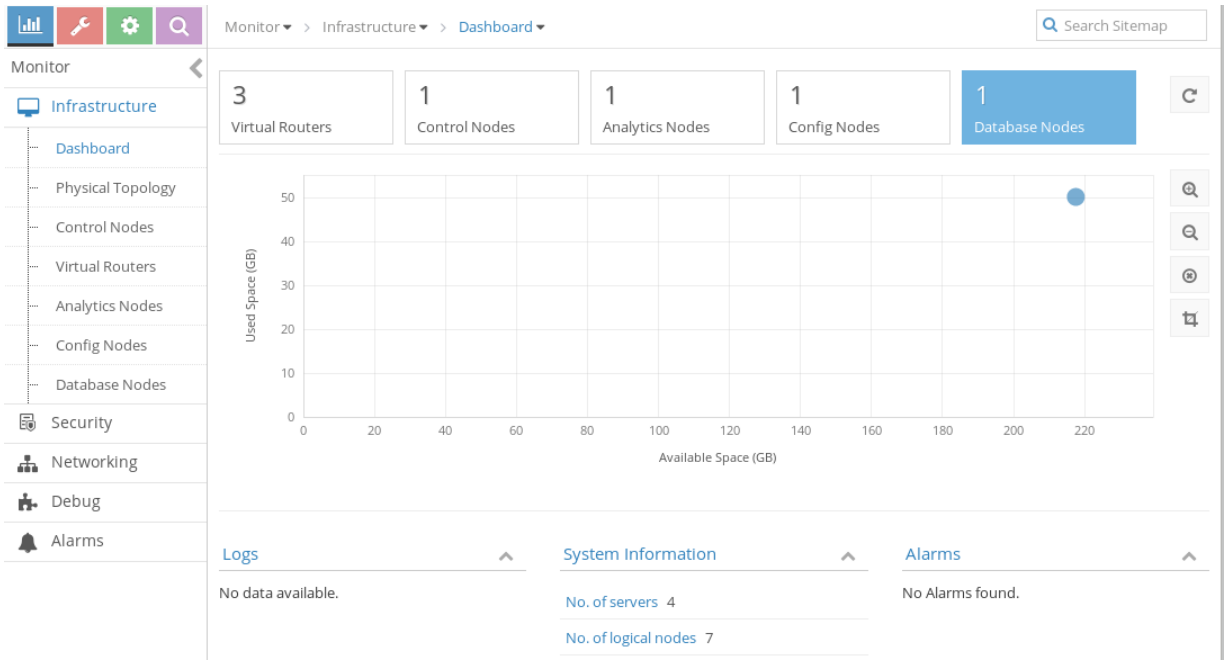
Figure 105 on page 261 displays the Contrail Web UI dashboard with all optional analytics components deployed.

Figure 105: Web UI - All Optional Analytics Components Deployed



A database node is visible in the infrastructure dashboard.

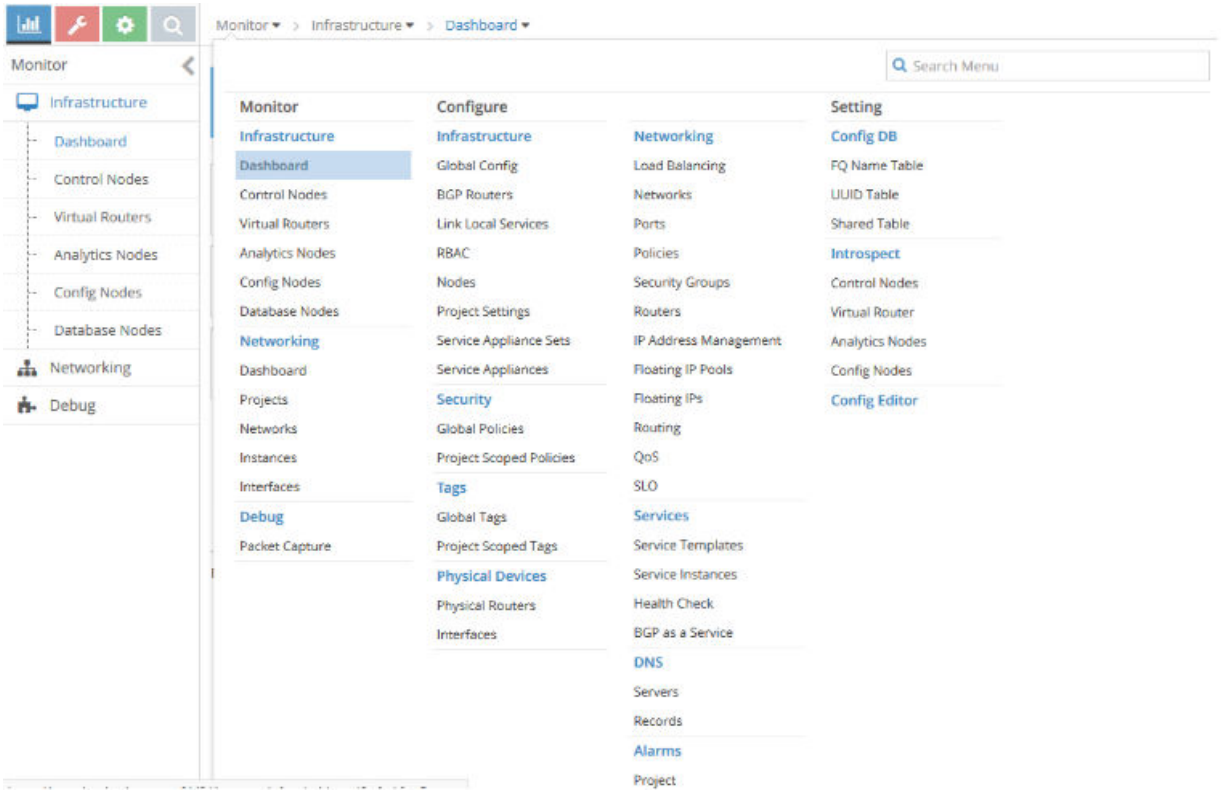
Figure 106: Web UI - Database Node in Dashboard



Web UI without Optional Components

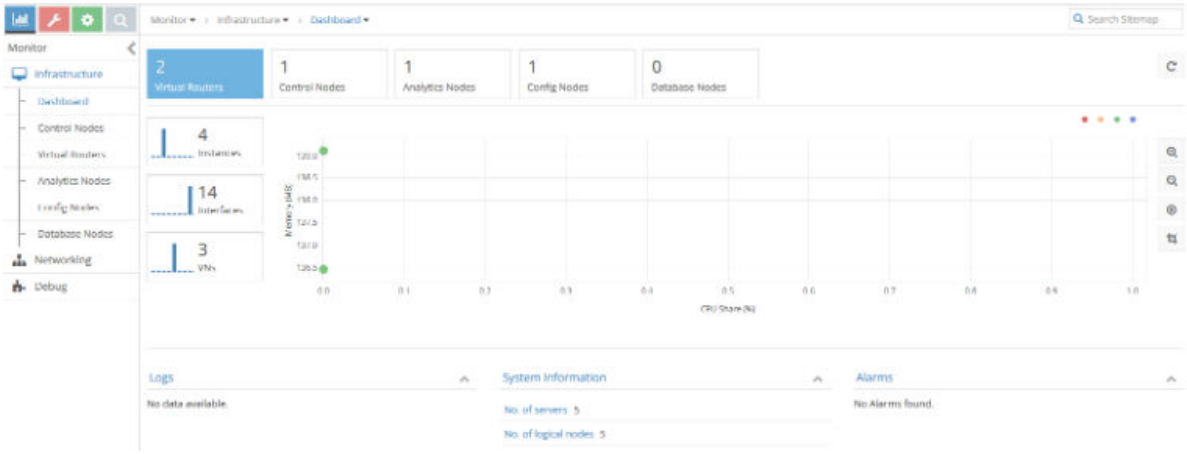
Figure 107 on page 263 displays the Contrail Web UI dashboard without optional analytics components deployed.

Figure 107: Web UI - Optional Analytics Components Not Deployed



No database node is visible in the infrastructure dashboard:

Figure 108: Web UI - Database Node Not Visible in Dashboard



Analytics Alarm Feature Enabled

Figure 109 on page 264 displays the **Monitor > Alarms** menu.

Figure 109: Web UI - Monitor > Alarms Menu

Figure 110 on page 264 displays the **Configure > Alarms** menu.

Figure 110: Web UI - Configure > Alarms Menu

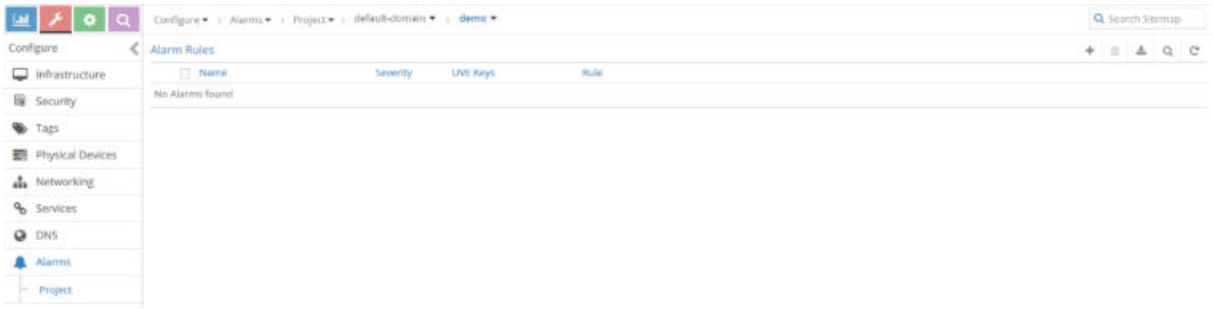
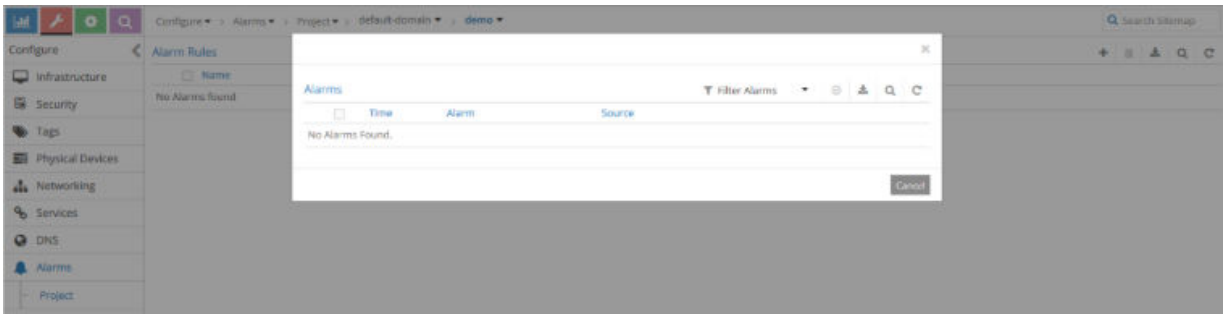


Figure 111 on page 265 displays the dialog box which appears when **Global Alarm**, next to Logged in User in the upper right, is selected.

Figure 111: Web UI - Global Alarm Settings



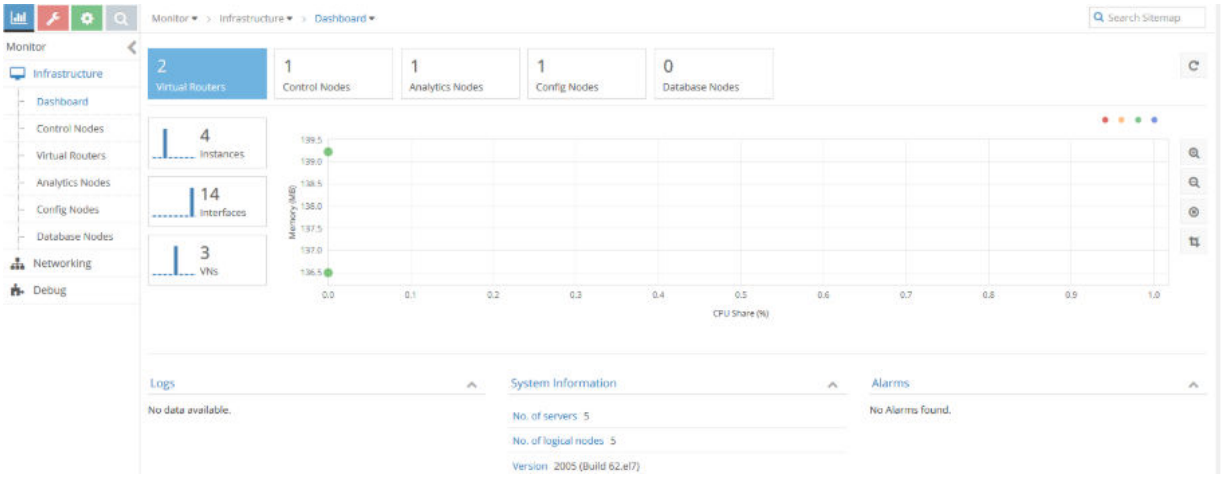
Analytics Alarm Feature Disabled

If the alarm analytics component is not deployed, then Contrail Web UI should not display the following alarm references:

- Global Alarm (Next to Logged in User)
- Monitor > Alarms
- Configure > Alarms

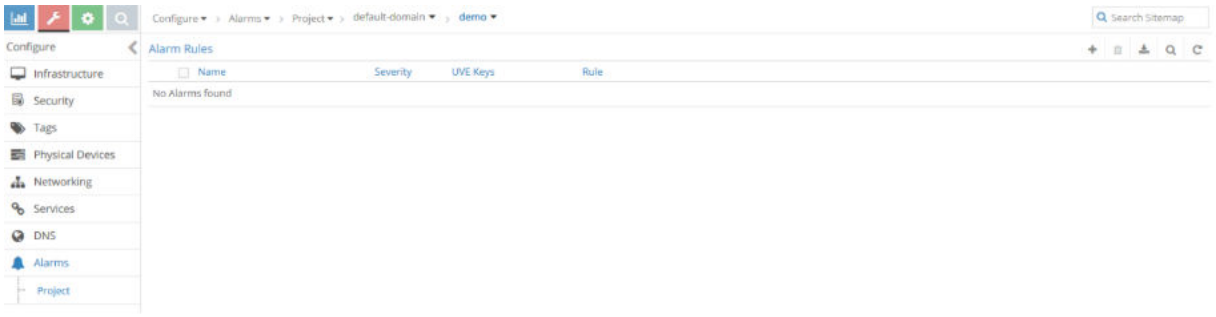
There is not an appearance of Global Alarm or **Alarms** entry in the Monitor menu:

Figure 112: Analytics Alarm Disabled - Global Alarm and Alarm Not Available



Alarms menu still available in Configure menu.

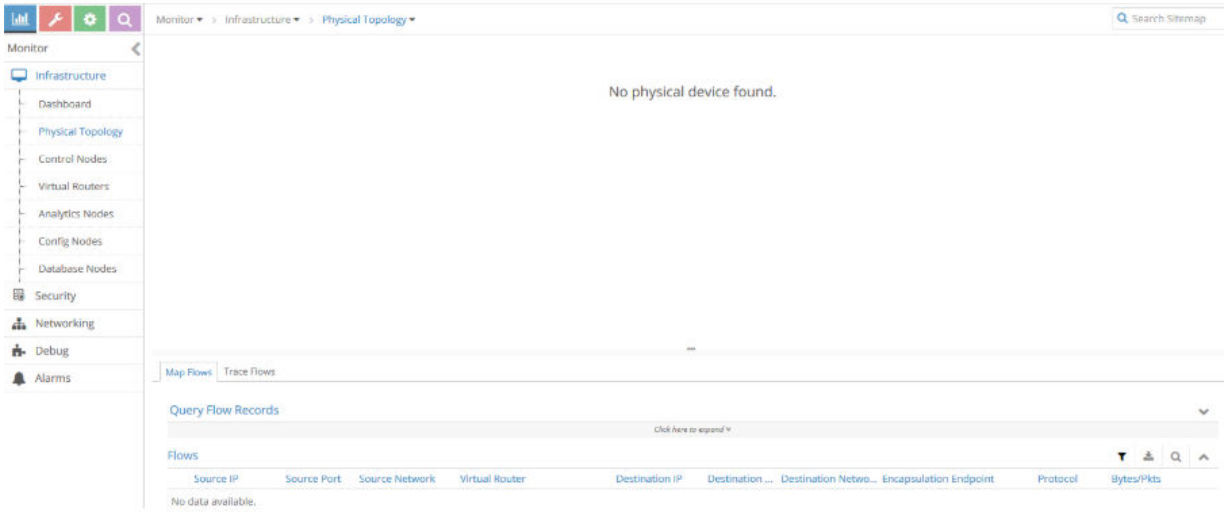
Figure 113: Analytics Alarm Disabled - Configure > Alarms



Analytics SNMP Feature Enabled

Figure 114 on page 266 displays the Physical Topology option in the Monitor menu.

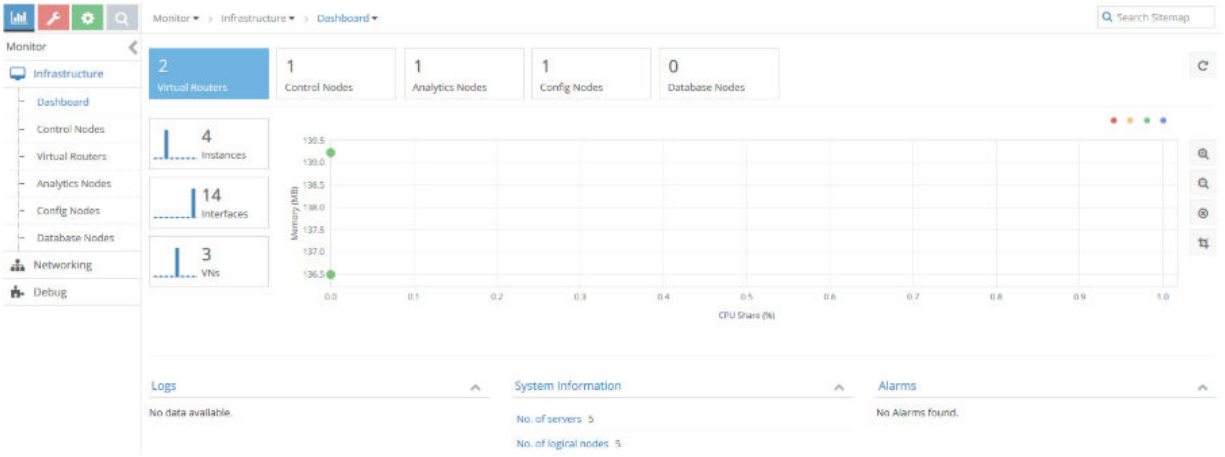
Figure 114: Analytics SNMP Feature Enabled - Physical Topology Menu Available



Analytics SNMP Feature Disabled

If the alarm analytics component is not provisioned, then Contrail Web UI does not display the Physical Topology menu option.

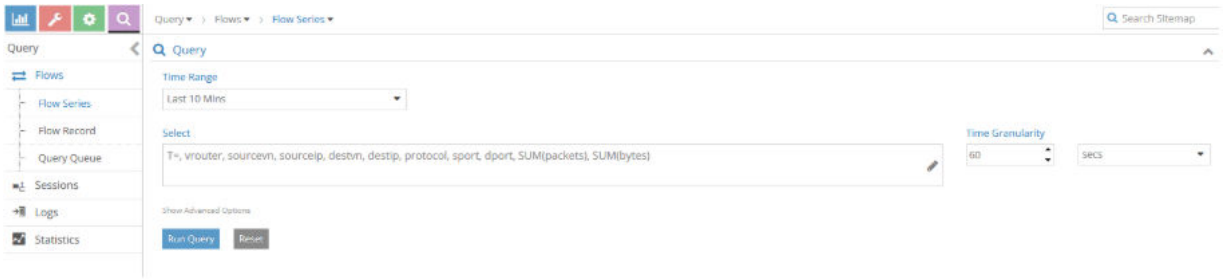
Figure 115: Analytics SNMP Feature Disabled - Physical Topology Menu Not Available



Analytics Database Enabled

If analytics database is provisioned, then Contrail Web UI displays the Query page.

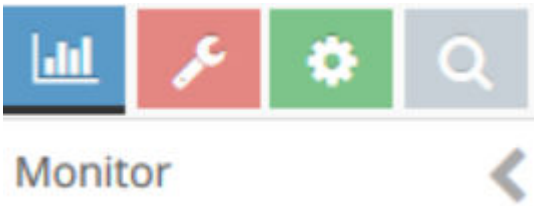
Figure 116: Analytics Database Enabled - Query Page Available



Analytics Database Disabled

If analytics database is not provisioned, then Contrail Web UI should not display the Query page. Query page logo is unavailable to launch Query page.

Figure 117: Analytics Database Disabled - Query Page Logo Not Available



Tripleo Provisioning

Multi-Nodes Contrail Controller Topology

In order to enable or disable the Contrail analytics optional components, TripleO templates have to be modified.

- In ContrailAnalytics role, **ContrailAnalyticsSnmp**, and **ContrailAnalyticsAlarm** resources can be removed:

```
- OS::TripleO::Services::ContrailAnalytics
- OS::TripleO::Services::ContrailAnalyticsSnmp
- OS::TripleO::Services::ContrailAnalyticsAlarm
```

- ContrailAnalyticsDatabase role can also be removed (not selected using ContrailAnalyticsDatabaseCount = 0) into a rollout as this role is deploying only **ContrailAnalyticsDatabase** resource:

```
- OS::TripleO::Services::ContrailAnalyticsDatabase
```

- ContrailController role is kept unchanged.

Single Node Contrail Controller Topology

In order to enable or disable the Contrail analytics optional components, TripleO templates have to be modified. In ContrailController role, **ContrailAnalyticsSnmp**, **ContrailAnalyticsAlarm**, and **ContrailAnalyticsDatabase** resources can be removed, other contrail resources are kept:

```
- name: ContrailController
    - OS::TripleO::Services::ContrailAnalytics
    - OS::TripleO::Services::ContrailAnalyticsAlarm
```

```
    - OS::TripleO::Services::ContrailAnalyticsDatabase
    - OS::TripleO::Services::ContrailAnalyticsSnmp
    - OS::TripleO::Services::ContrailCertmongerUser
    - OS::TripleO::Services::ContrailConfig
    - OS::TripleO::Services::ContrailConfigDatabase
    - OS::TripleO::Services::ContrailControl
    - OS::TripleO::Services::ContrailWebui
```

TripleO Template Update

TripleO templates were updated in June 2020 to allow disabling the provisioning of Contrail analytics components.

Earlier Contrail TripleO templates have to be patched in order to replace `docker/services/contrail/contrail-base.yaml` file in which optional analytics component provision is hardcoded:

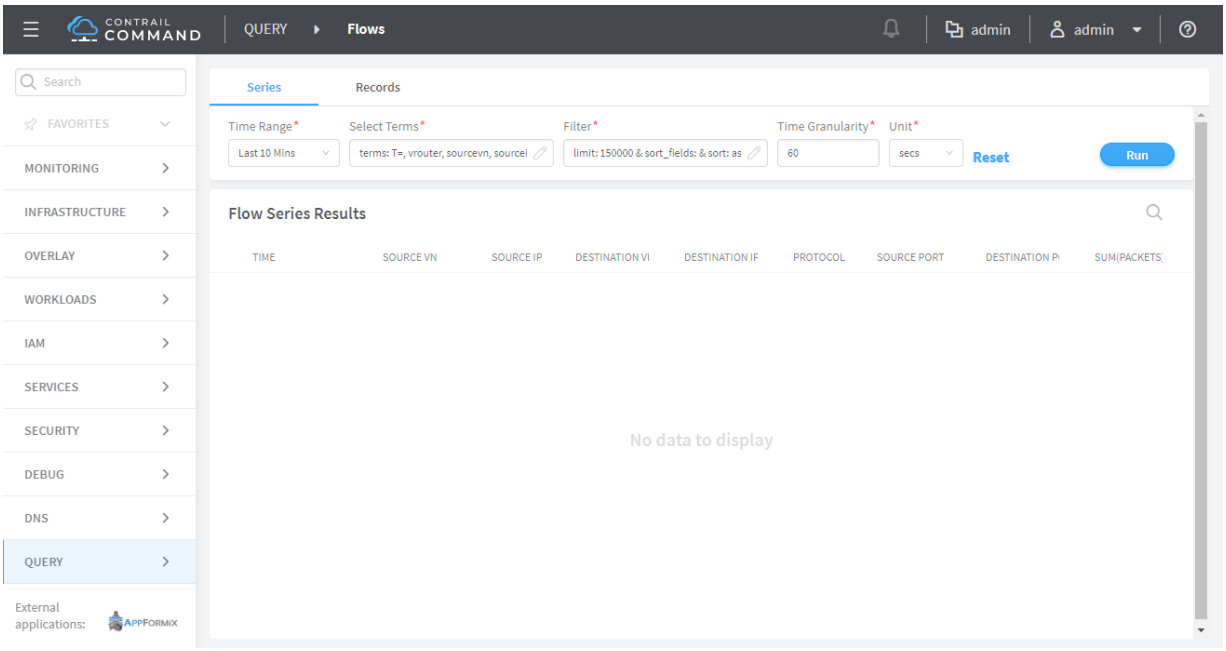
```
...
                ANALYTICS_ALARM_ENABLE: 'False'
                ANALYTICS_SNMP_ENABLE: 'True'
                ANALYTICSDB_ENABLE: 'True'
...
```

Appendix

Contrail Command UI

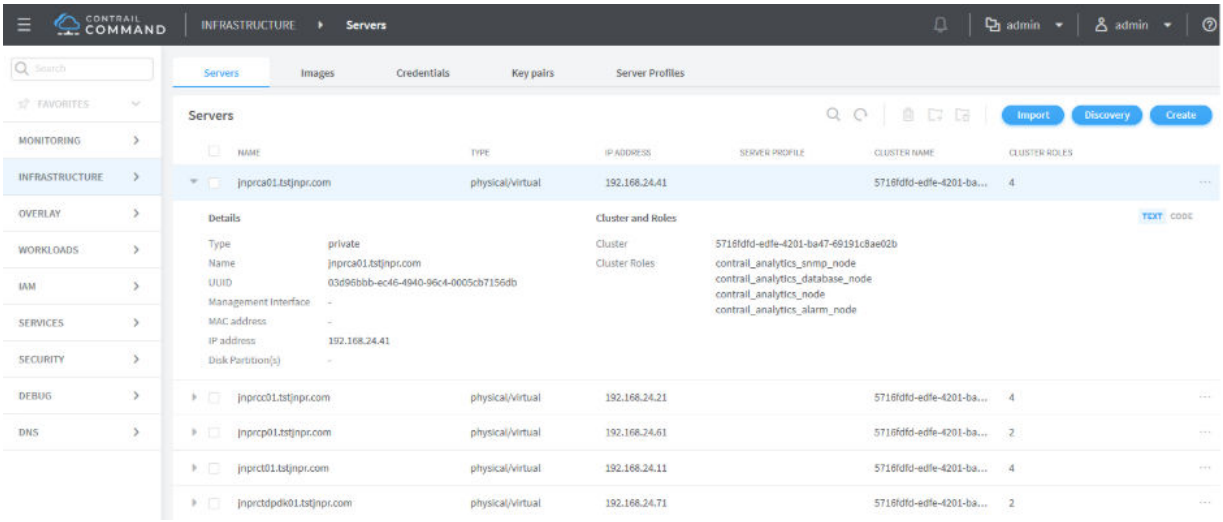
The disabled roles and charts are visible on the Query page but they are not operational.

Figure 118: Query Page Visible in Dashboard



Regardless that the alarm, SNMP, and database analytics roles have been disabled, they are still reported by Contrail Command.

Figure 119: Disabled Roles Still Visible in Contrail Command



The following five charts will always display empty.

Figure 120: Empty Charts in Compute Nodes

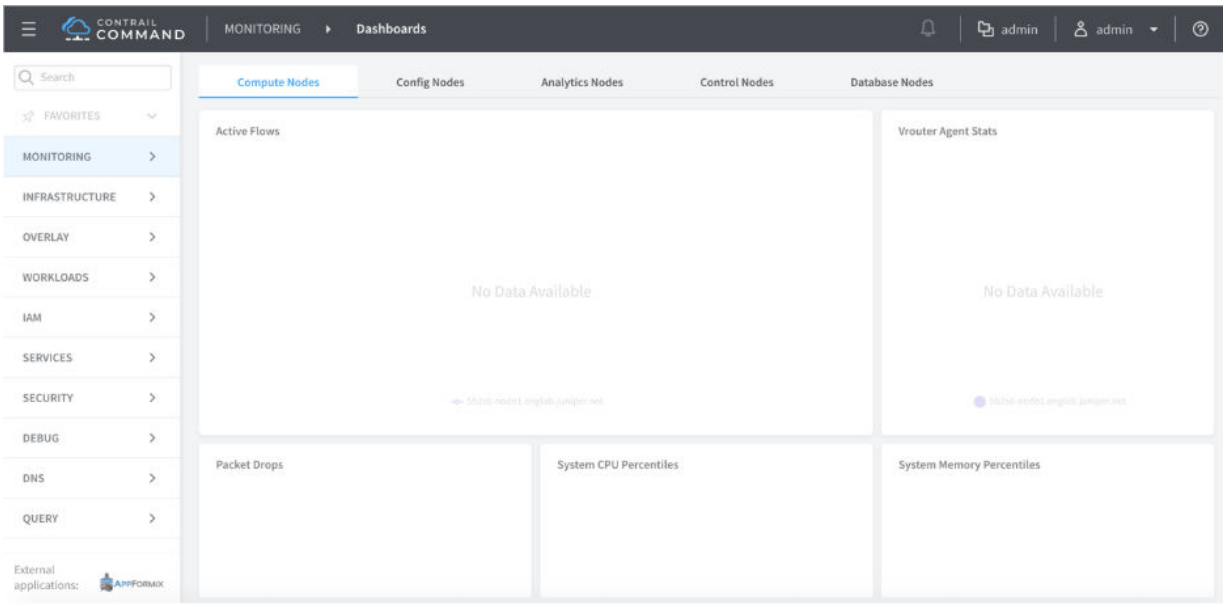


Figure 121: Empty Charts in Config Nodes

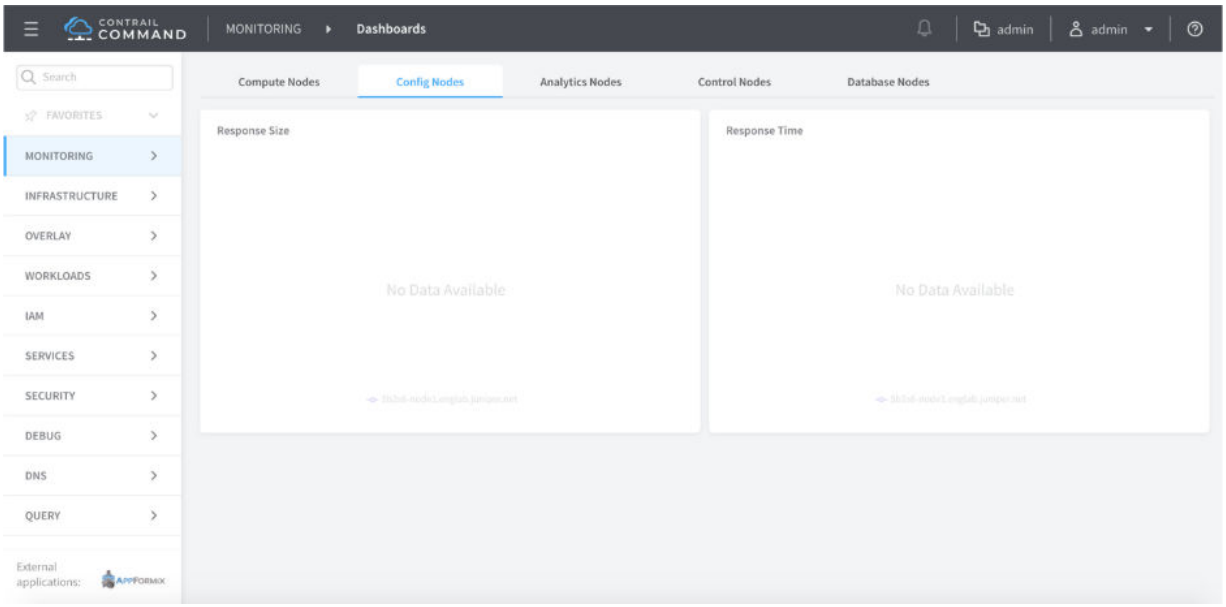


Figure 122: Empty Charts in Analytics Nodes

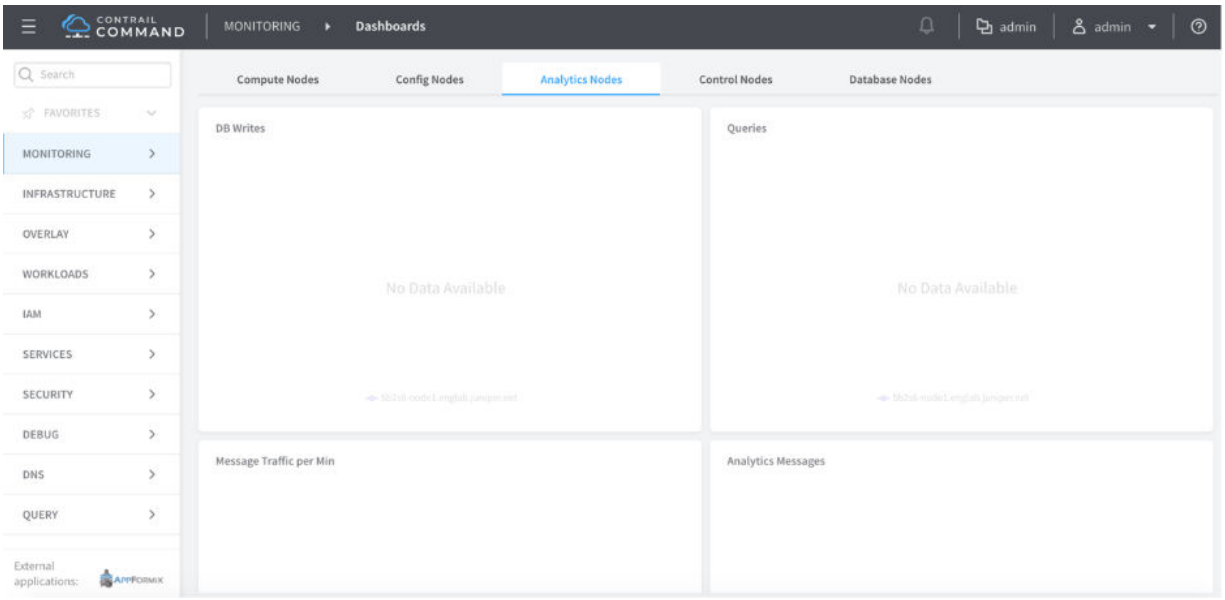


Figure 123: Empty Charts in Control Nodes

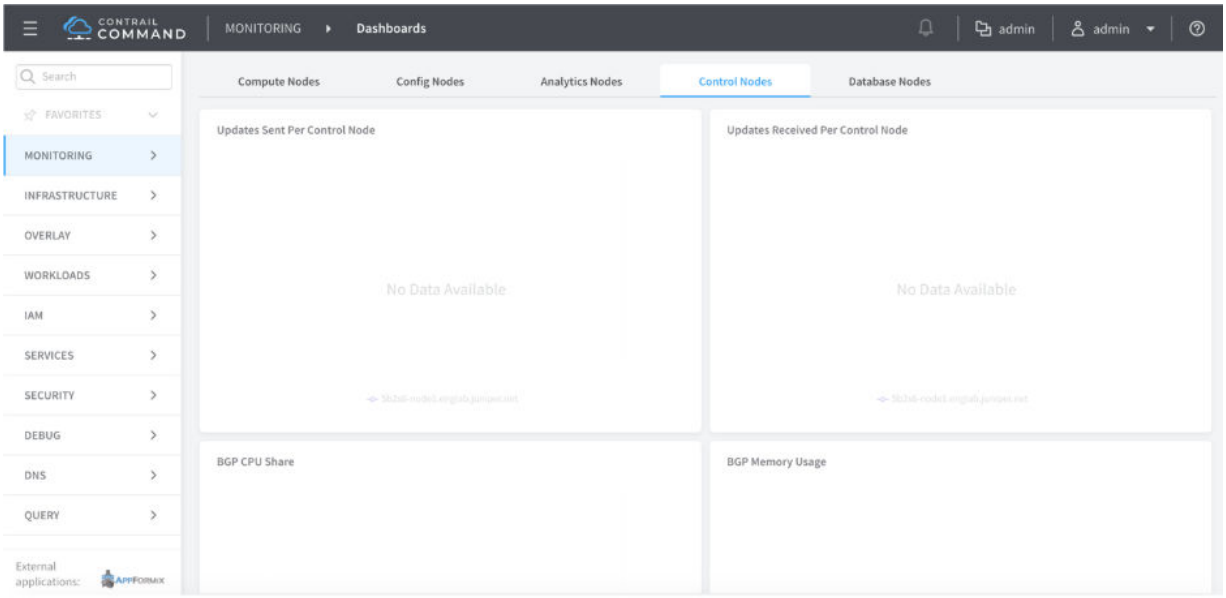
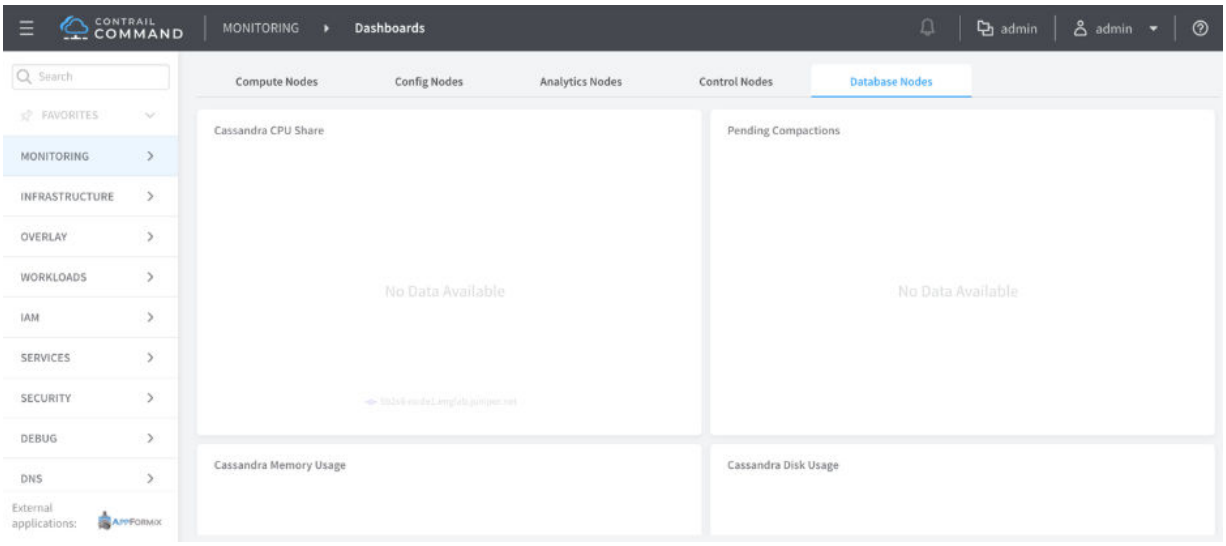
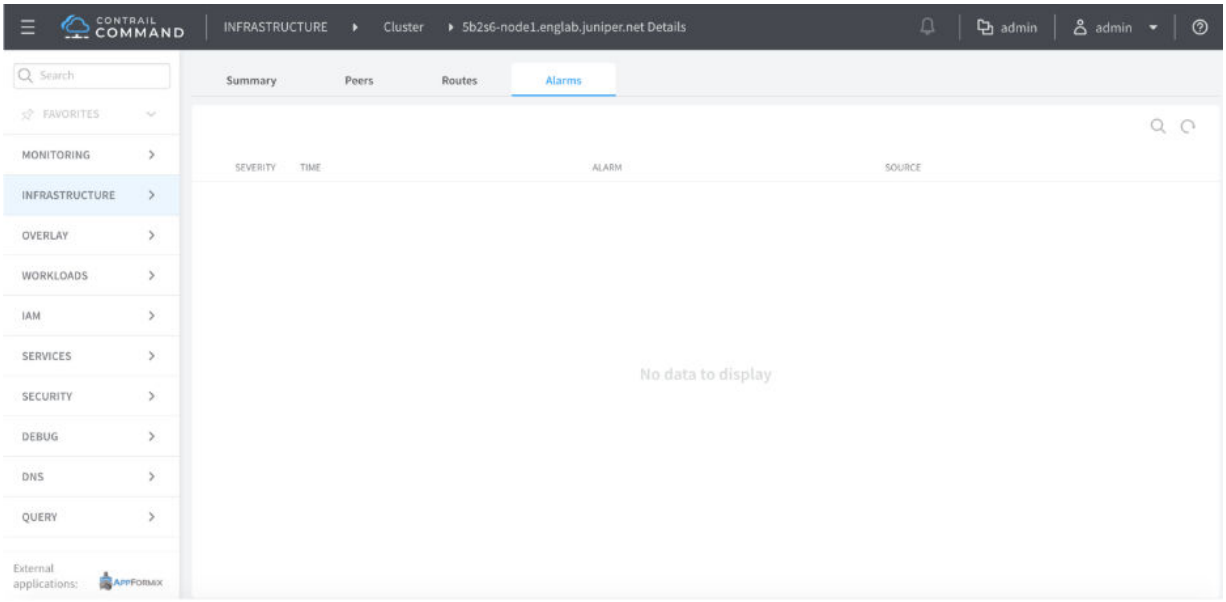


Figure 124: Empty Charts in Database Nodes



The Alarms page displays alarms pulled from the Contrail `analytics_alarm` component. When the `analytics_alarm` component is disabled, the Alarms page will always display no data.

Figure 125: Empty Alarms Page



Change History Table

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

Release	Description
2008	TripleO templates were updated in June 2020 to allow disabling the provisioning of Contrail analytics components.

CHAPTER 5

Using Contrail Command to Monitor and Troubleshoot the Network

IN THIS CHAPTER

- Viewing Overlay Routes | 275
- Monitoring Bond Interfaces in DPDK Enabled Devices | 276
- Top N View in Contrail Command | 280
- Viewing Topology Maps from Contrail Command | 286
- Viewing Packet Path in Topology View | 291
- Assign Custom Names to Privileged Ports and VXLAN IDs | 296
- Viewing the Monitoring Dashboards | 302
- Creating a Query for Flows | 306
- Contrail Analytics Optional Modules | 315

Viewing Overlay Routes

Contrail Networking Controller peers with Multi-Protocol BGP (MP-BGP) routers with both data center devices, such as underlay switches (leaf and spine), as well as SDN gateways. The controller receives and advertises the routes through control nodes. Starting with Contrail Networking Release 1910, you can view, filter and search the overlay MP-BGP routes on Contrail Command.

Being able to view control node data enables you to debug and troubleshoot networking issues.

To view the overlay routes, perform the following steps.

- Navigate to the **Infrastructure > Cluster** page. The **Overview** tab is displayed with an overview of the cluster infrastructure components, including the numbers of control nodes, compute nodes, analytics nodes, config nodes, and database nodes currently operational and also virtual networks. You can also view charts displaying config nodes response sizes against response time as well as analytics message sizes against time.

- Click **Cluster Nodes** to view more details on each of these nodes. The **Cluster *Name* Nodes** page appears with lists of control nodes, compute nodes, service nodes, multicloud gateway nodes, and baremetal servers in their separate respective tabs.
- To view more details on control nodes, select a control node on the **Control Nodes** tab. The ***Control_Node* Details** page appears.
- The ***Control_Node* Details** page has multiple tabs.
  - The **Summary** tab provides a summary of the status and activity on the selected node. It also displays charts detailing host CPU usage, memory usage, and so on. In the Contrail Web UI, similar information was available under **Monitor > Infrastructure > Control Nodes > Summary**. However, Contrail Command displays more details since it uses Contrail Insights to generate the data.
  - Click the **Peers** tab to view information about peers established for this control node. The **Peers** tab displays the peers for an individual control node and their peering state. Click the ▸ icon next to the peer name to expand, view, and copy peer information. You can use the search field on the top right of the page to search for peers based on specified input strings.
  - Click the **Routes** tab to view information on the routes. The **Routes** tab displays active routes for this control node and lets you query the results. Click the filter icon on the top right of the page to apply filters while searching for routes. You can also apply multiple filters.
  - Click the **Alarms** tab to view all alarms on the control node. Click the ▸ icon next to the alarm name to expand and view alarm details.

Change History Table

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

Release	Description
1910	Starting with Contrail Networking Release 1910, you can view, filter and search the overlay MP-BGP routes on Contrail Command.

Monitoring Bond Interfaces in DPDK Enabled Devices

Starting with Contrail Networking Release 1910, you can use the Contrail Command user interface (UI) to monitor the status of primary and secondary devices that are members of a bond interface. This feature is available for device systems configured with Data Plane Development Kit (DPDK). In releases prior to release 1910, you could only see a bond interface in the Contrail Command UI. In release 1910,

you can view the details and receive notifications about the status of the primary as well as secondary devices in the bond interface.

To monitor the members in a bond interface, perform the following steps:

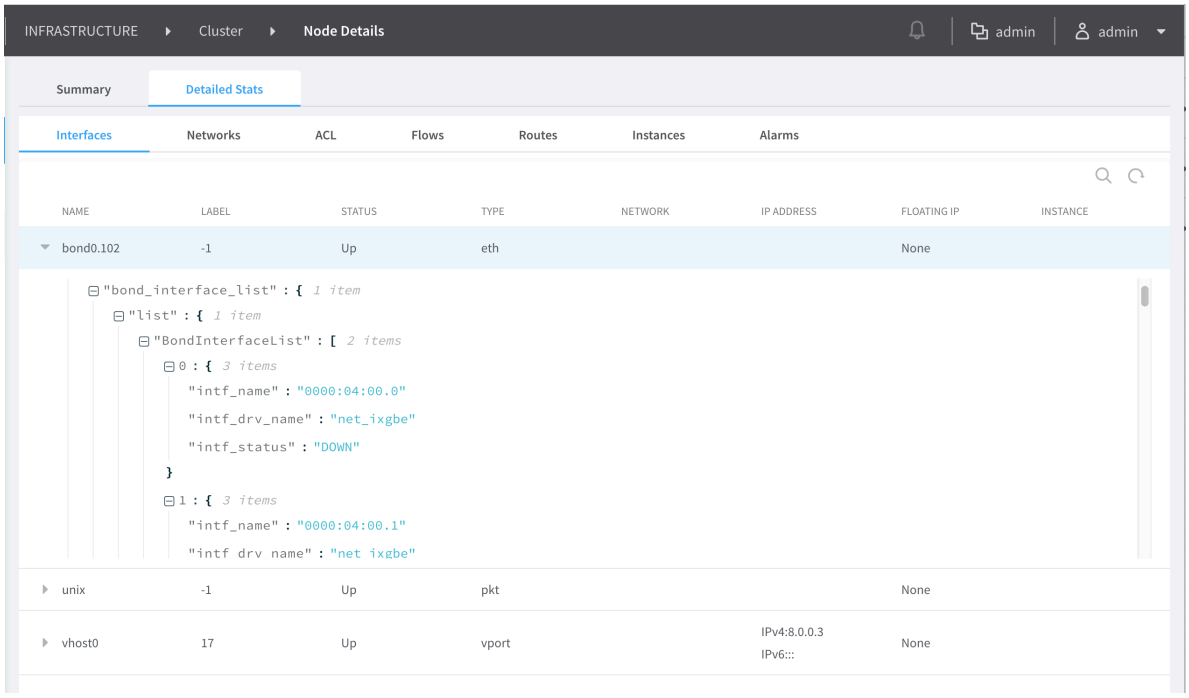
1. Click **Infrastructure**>**Cluster**.

   The **Overview** page is displayed.
2. Click **Cluster Nodes**. The **Cluster AIO Nodes** page is displayed.

   Click **Compute Nodes** tab. A list of nodes is displayed.
3. Click any node to view the *Node Details*. The **Summary** tab is displayed.
4. Click **Interfaces** tab. A list of interfaces deployed in the node is displayed.

   Click on a bond interface to view and monitor the bond interface. See Figure 126 on page 277 .

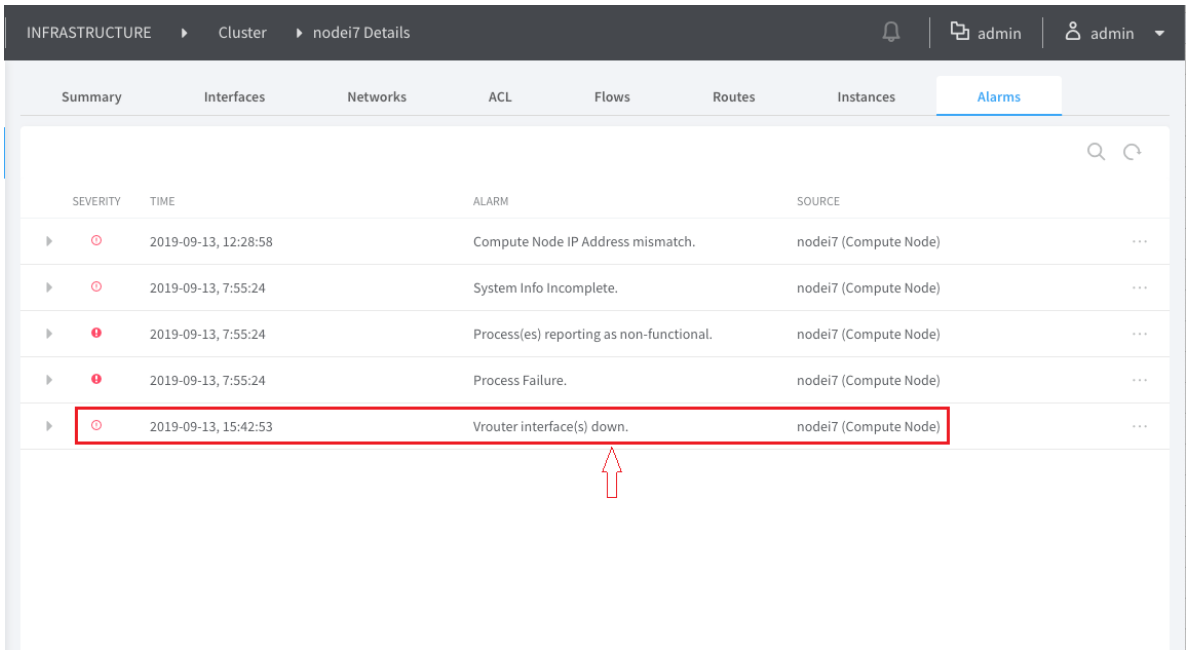
Figure 126: Members of the Bond Interface



5. Click **Alarms** tab. A list of alarms is displayed.

   This tab overrides the old alarms and shows you the latest alarm generated when the primary or secondary device in the bond interface goes **DOWN**. This indicates that the member has become inactive. See Figure 127 on page 278 .

Figure 127: Alarms Generated by Bond Interface Members



NOTE: In a multi-node setup, when the primary interface goes down in a DPDK enabled device, the Contrail Command UI cannot display the status as the connection between the controller and the primary interface is inactive. The Contrail Command UI obtains the previous status from the cache and displays it.

You can also use `vif-list` command on the CLI to view the details of the bond interface members.

Executing the `vif-list` command gives you the following output when all interface members are **UP**:

```
vif0/0      PMD: 0 (Speed 1000, Duplex 1)
                 Type:Physical HWaddr:9e:b1:2a:68:e8:58 IPaddr:0.0.0.0
                  Vrf:0 Mcast Vrf:65535 Flags:XTcL3L2VpDpdk QOS:0 Ref:19
                 RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                 Fabric Interface: eth_bond_bond0  Status: UP  Driver: net_bonding
                 Slave Interface(1): 0000:04:00.0  Status: UP  Driver: net_ixgbe
                 Slave Interface(2): 0000:04:00.1  Status: UP  Driver: net_ixgbe
                 RX packets:0  bytes:0 errors:0
                 TX packets:5  bytes:430 errors:0
                 Drops:0
```

```
                    TX port   packets:5 errors:0
                    TX device packets:5  bytes:450 errors:0
```

Executing the `vif-list` command gives you the following output when all interface members are **DOWN**:

```
vif0/0      PMD: 0 (Speed 1000, Duplex 1)
                    Type:Physical HWaddr:9e:b1:2a:68:e8:58 IPaddr:0.0.0.0
                    Vrf:0 Mcast Vrf:65535 Flags:XTcL3L2VpDpdk QOS:0 Ref:19
                    RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                    Fabric Interface: eth_bond_bond0  Status: DOWN  Driver: net_bonding
                    Slave Interface(1): 0000:04:00.0  Status: DOWN  Driver: net_ixgbe
                    Slave Interface(2): 0000:04:00.1  Status: DOWN  Driver: net_ixgbe
                    RX packets:0  bytes:0 errors:0
                    TX packets:5  bytes:430 errors:0
                    Drops:0
                    TX port   packets:5 errors:0
                    TX device packets:5  bytes:450 errors:0
```

Executing the `vif-list` command gives you the following output when bond interface is not configured and there are no secondary devices:

```
vif0/0      PMD: 0 (Speed 1000, Duplex 1)
                    Type:Physical HWaddr:9e:b1:2a:68:e8:58 IPaddr:0.0.0.0
                    Vrf:0 Mcast Vrf:65535 Flags:XTcL3L2VpDpdk QOS:0 Ref:19
                    RX queue errors to lcore 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                    Fabric Interface: 0000:04:00.0  Status: DOWN  Driver: net_ixgbe
                    RX packets:0  bytes:0 errors:0
                    TX packets:5  bytes:430 errors:0
                    Drops:0
                    TX port   packets:5 errors:0
                    TX device packets:5  bytes:450 errors:0
```

Change History Table

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

Release	Description
1910	Starting with Contrail Networking Release 1910, you can use the Contrail Command user interface (UI) to monitor the status of primary and secondary devices that are members of a bond interface.

RELATED DOCUMENTATION

vRouter Command Line Utilities | 34

Top N View in Contrail Command

SUMMARY

This topic covers the Top N feature in the Contrail Command GUI. Contrail Insights has a new Top N or “top talkers” query engine with tabular and charted views. These Contrail Insights diagnostics enable engineers to proactively mitigate issues like network congestion and resource contention.

IN THIS SECTION

- Contrail Command UI—Top N Feature | 280
- Top N Filter Options | 282
- Chart View | 285

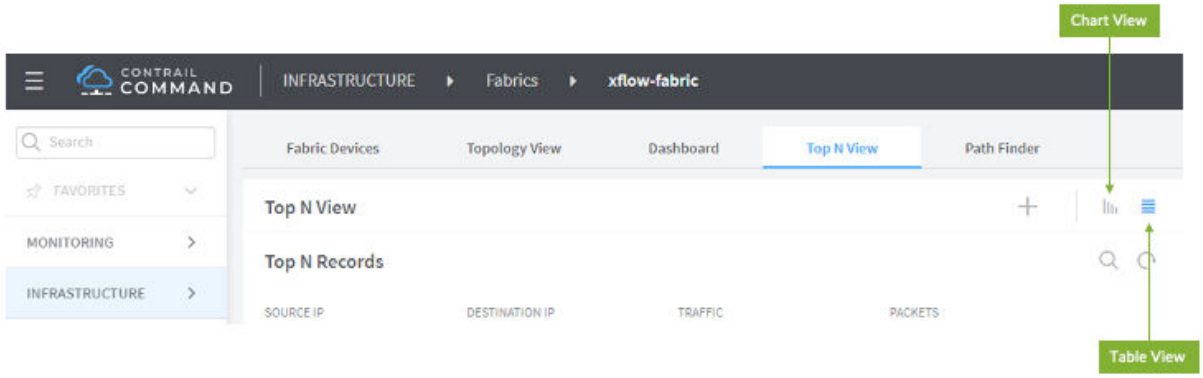
Contrail Command UI—Top N Feature

Follow the steps to navigate to the **Top N View**.

1. Log in to a cluster via Contrail Command by browsing to https://<Contrail-Command-Server-IP-Address>:9091.
2. Navigate to **Infrastructure** > **Fabrics**.
3. Select the desired **Fabric** from the available list.
4. Click on **Top N View**.

The feature offers *table view* and *chart view*.

The default view is the *table view*.



In addition to the standard Top N results, you can also define custom Top N fields to group the results. Click on the **+** button to group the results based on different attributes. You can add or remove the desired attributes. The following grouping values are available:

- Packets
- Average speed
- Average number of packets
- Average size of packets
- Source IP
- Destination IP
- Source Port
- Destination Port
- Protocol
- Source Virtual Network
- Destination Virtual Network
- Overlay Source IP
- Overlay Destination IP
- Overlay Source Port
- Overlay Destination Port
- Overlay Protocol
- Network Device
- Source Interface
- Destination Interface
- IP Version
- IP Tos
- Source ASN
- Destination ASN
- Source Network Mask

- Destination Network Mask
- Source MAC Address
- Destination MAC Address

Top N Filter Options

The top–N results show the Top N contributors or “top talkers” to the network traffic and how the Top N contributors change over time. These results are generated from the sampled packets exported by sFlow.

The following network traffic Top N options are available—

Top N Options	Description	Default Values
Predefined Time	Select the period in the history for which data is to be displayed.	
Time range	Use the calendar or type directly into the fields to select the desired start and end time. Additionally, you can select a time interval by dragging the mouse.	
Network Device	Filter data passing through the network device	
Source Interface	Filter the source interface on the selected network device	
Destination Interface	Filter the destination interface on the selected network device	
N Records	Select the number of records for the top talkers of traffic	15
Include Missing	Enable to see the results including traffic between physical devices that are not related to overlay	Disabled
Deduplication	Enable to see the results for the actual scale of traffic transferred between source IP and destination IP. It eliminates counting the duplicate traffic reported by multiple network devices.	Disabled

(Continued)

Top N Options	Description	Default Values
Overlay Network		
Source Virtual Network	Filter data with this source virtual network	
Destination Virtual Network	Filter data with this destination virtual network	
Source IP/Mask	Filter data with the source IP or in the subnet range with mask	
Destination IP/Mask	Filter data with the destination IP or in the subnet range with mask	
Source Port	Filter data with the source port	
Destination Port	Filter data with the destination port	
Protocol	Filter data with the protocol type	Available Options: • ICMP • TCP • UDP
Underlay Network		
Source IP/Mask	Filter data with the source IP or in the subnet range with mask	
Destination IP/Mask	Filter data with the destination IP or in the subnet range with mask	
Source Port	Filter data with the source port	

(Continued)

Top N Options	Description	Default Values
Destination Port	Filter data with the destination port	
Protocol	Filter data with the protocol type	Available Options: • ICMP • OSPFIGP • TCP • UDP
Additional filters		
IP Version	Filter data with the IP Version	Available Options: • IPv4 • IPv6
IP Tos	Filter data with the IP type of service	
Source ASN	Filter data with the source autonomous system number (ASN)	
Destination ASN	Filter data with the destination autonomous system number (ASN)	
Source Net	Filter the data with source network	
Destination Net	Filter the data with destination network	
Source MAC	Filter data with the source MAC address	
Destination MAC	Filter data with the destination MAC address	

(Continued)

Top N Options	Description	Default Values
Encapsulation type	Filter data with the encapsulation type	Available Options: • vxlan • mpls

Chart View

You can also analyze the Top N results in the *chart view*.

Click on the **Chart View** button on the **Top N View** page.

Click on the **Configure** button to customize the results.

The following types of charts are available:

- Bar Chart
- Pie Chart
- Donut Chart
- Treemap

You can customize the **Y-axis** based on—

- Bytes
- Packages
- Average Packet Size
- Average Speed
- Average Packet Number

You can also add heatmap for Y-axis parameters by enabling the option, **Custom heatmap**. It adds another dimension to the chart view. The option will sort the colors of the chart based on the selected heatmap parameter.

Select the desired fields from the **PARAMETERS DISPLAYED** list to group the results by various parameters. You can hover your mouse over the chart to see these parameters.

Click **Apply** to see the results.

RELATED DOCUMENTATION

Contrail Insights Flows in Contrail Command | 339

Viewing the Network Topology

Viewing Topology Maps from Contrail Command

IN THIS SECTION

- Filters | 291

You can view heatmaps from the **Infrastructure** > **Fabrics** > *Fabric Name* > **Topology View** page.

The Topology View feature helps visualize the heatmaps generated based on the metrics collected through sources such as sFlows and SNMP. You can select a time range from the **Summary** section on the left panel to view the network statistics during a specific time period.

The metrics are represented in color and with a temperature scale displayed on the right of the topology as shown in Figure 128 on page 287 . The maximum numeric value retrieved for the metric among all possible nodes or edges are represented as red on the scale. The lowest possible value, 0 (zero), is represented as Green. The color gradation is from Red to Green with Yellow at 50%. For metrics represented in percentages, the maximum (100%) is represented as Red, yellow is 50% and green is 0%. For a node, the heat color is based on the maximum of the sum of either ingress or egress traffic.

Figure 128: Topology View

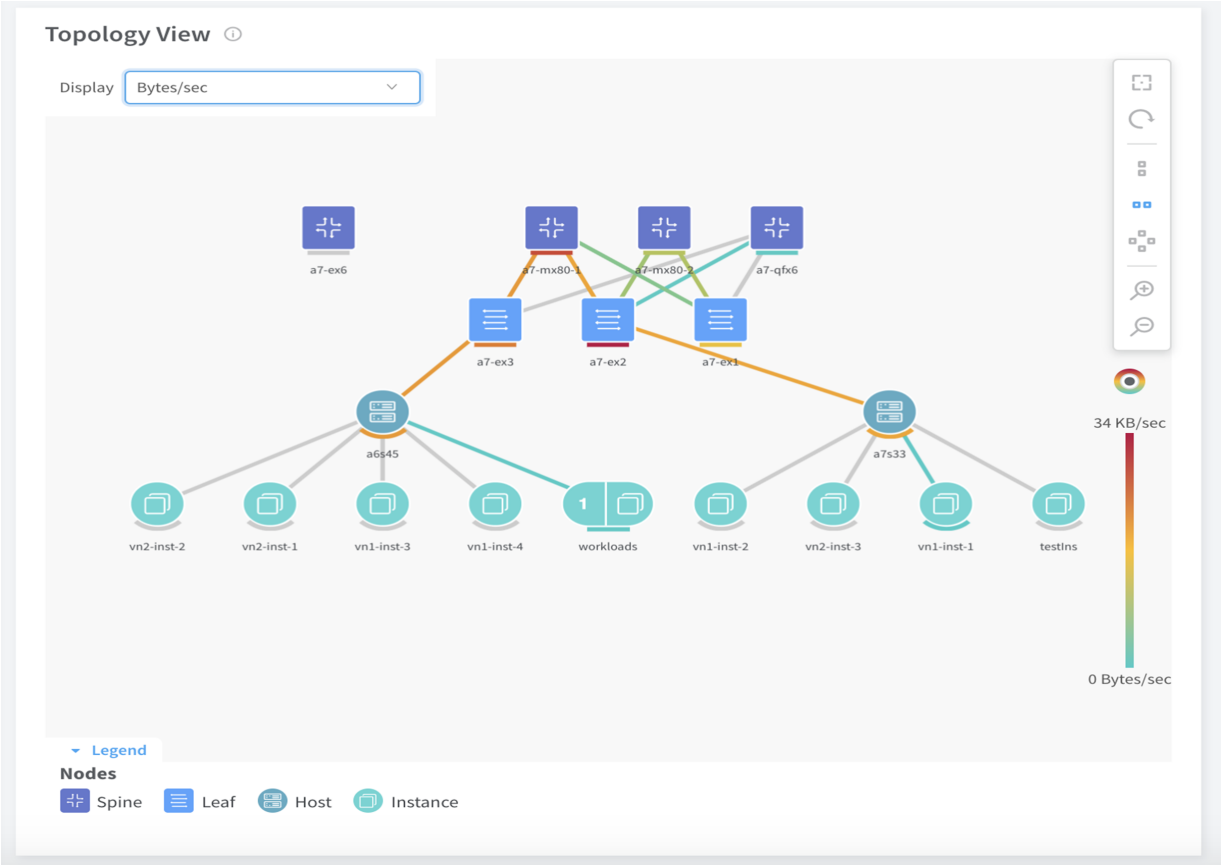
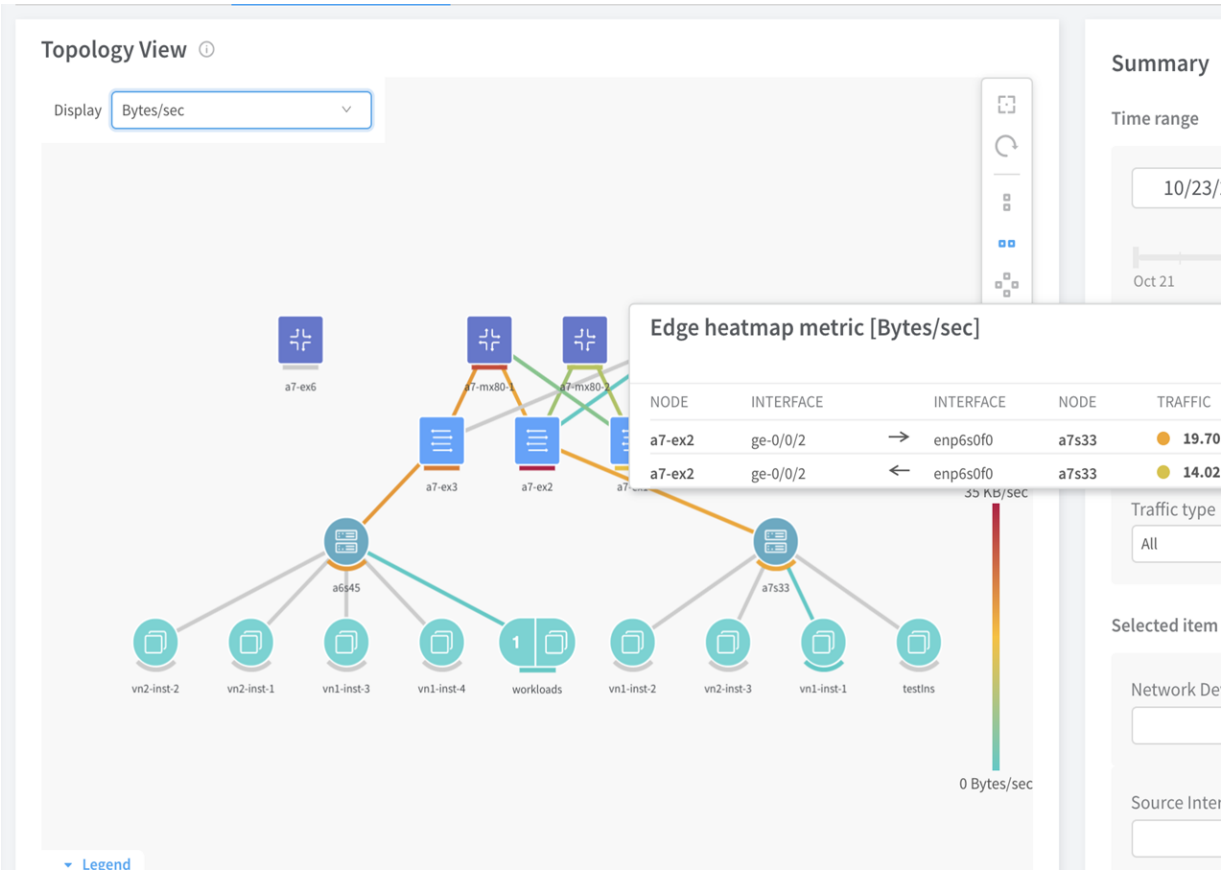


Figure 128 on page 287 shows the Bytes/sec metric to visualize the heatmap. The temperature scale on the right indicates that 34 Kb/sec is the maximum rate discovered among all devices and is color coded in red. The lowest value is 0 Bytes/sec, which is shown as green. You can hover on the edge devices or nodes to view the actual values as shown in Figure 129 on page 288 .

Figure 129: Edge Devices Heatmap Metric



The node color for a7-ex2 is based on the values of the maximum of sum of ingress versus the egress traffic.

For example, in Figure 130 on page 289 , the sum of egress traffic is 13.32 KB/sec and ingress is traffic is 12.30 KB/sec. Hence, for this node, the traffic is shown as 13.21 KB/sec and is marked red. This is because this value is close to the maximum values obtained from all the links and the nodes.

Figure 130: Node Heatmap Metric

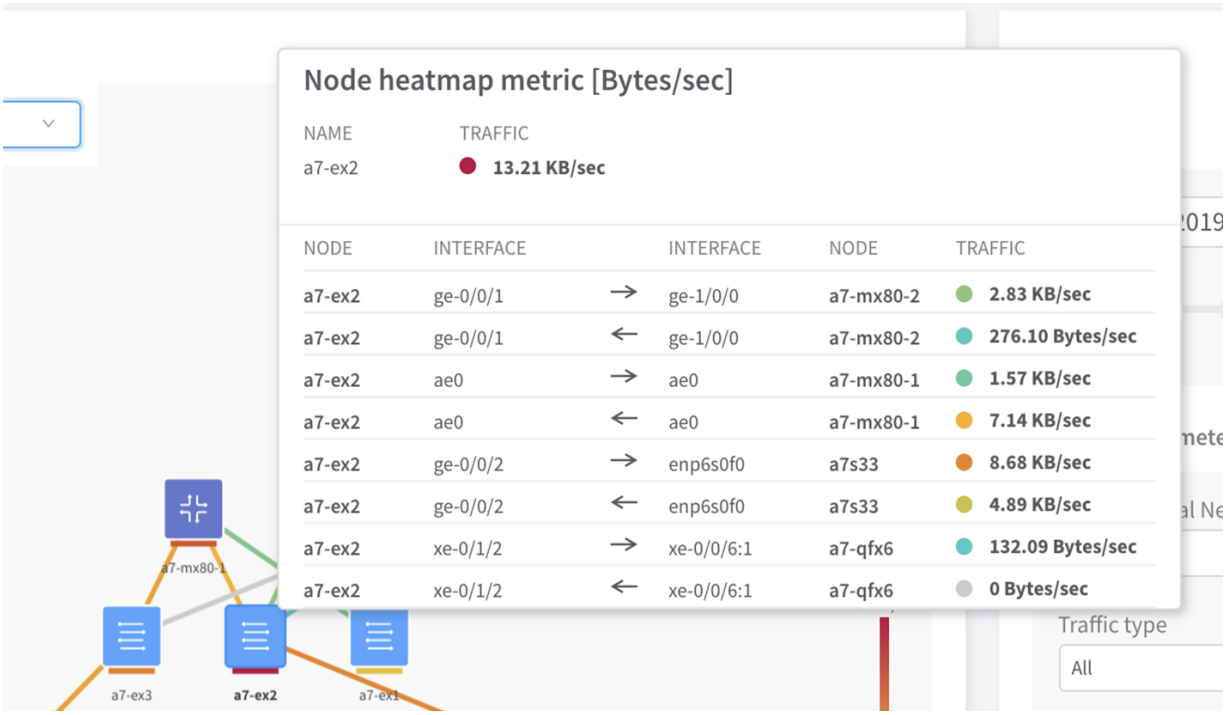


Figure 131 on page 289 shows the sample traffic data collected and provided by the Contrail Insights Flows API.

Figure 131: Sample Traffic from Contrail Insights Flows API


```
▼traffic: {bytes: 967017600, packets: 2823520, utilization: 0.00004476933333333333}
   bytes: 967017600
   packets: 2823520
   utilization: 0.00004476933333333333
```


Table 48 on page 289 describes the metrics, its source, and the method of calculation.

Table 48: Metrics and Its Source

Metric	Source/Collector	Calculation Method
Bytes/sec	sFlow / Contrail Insights Flows	Calculated by dividing the sum of bytes by time range in seconds.

Table 48: Metrics and Its Source *(Continued)*

Metric	Source/Collector	Calculation Method
Packets/sec	sFlow / Contrail Insights Flows	Calculated by dividing the sum of packets by time range in seconds.
Utilization	sFlow / Contrail Insights Flows	Utilization is % of link capacity used and is calculated based on the link capacity. The total utilization divided by time range in seconds is used to show average utilization over the time range.
Host/Instances—All host and instances metrics collected by Contrail Insights.		
Memory Usage	Contrail Insights	Percentage
CPU Usage	Contrail Insights	Percentage
Disk I/O Read	Contrail Insights	Average
Disk I/O Write	Contrail Insights	Average
Ingress Errors	Contrail Insights	Average
Egress Errors	Contrail Insights	Average
Ingress Drops	Contrail Insights	Average
Egress Drops	Contrail Insights	Average
User SNMP Metrics	Contrail Insights	All the SNMP metrics collected by Contrail Insights. For Contrail Insights to collect SNMP data, you need to configure SNMP and select IFMIB.

For more information, see Metrics Collected by Contrail Insights.

Filters

Clicking on a node or edge device filters data for the selected connection, network device, host, instance, or baremetal server. You can select the filter criteria from the **Heatmap parameters** section on the right pane. You can apply multiple filters at a time.

Table 49: Filter Parameters

Field	Description
Source Virtual Network	Filter traffic data for a particular source virtual network.
Destination Virtual Network	Filter traffic data for a particular destination virtual network.
Network Device	Filter traffic for the selected network device.
Host/Instance	Filter traffic data for the selected host or node instance.
Source Interface	Filter traffic for the selected source interface.
Destination Interface	Filter traffic for the selected destination interface.

RELATED DOCUMENTATION

Contrail Insights Flows in Contrail Command | 339

Viewing Packet Path in Topology View | 291

Viewing Packet Path in Topology View

IN THIS SECTION

- View Packet Paths | 292
- Fabric-Only Path Finding | 295

Starting in Contrail Networking Release 2008, you can view the path a packet takes in a network. Visualizing the packet path on a topology view alongside the traffic assists the administrator when troubleshooting. The visualization is possible by using the sFlow and topology data that a packet consumes traveling in a network. When the node and edge details are obtained, the topology is plotted and the path can be visualized in the topology view.

NOTE: To view the packet path, both Contrail Insights and Contrail Insights Flows must be installed. See *How to Install Contrail Command and Provision Your Contrail Cluster* in the *Contrail Networking Installation and Upgrade Guide*.

View Packet Paths

To view packet paths in the Topology View:

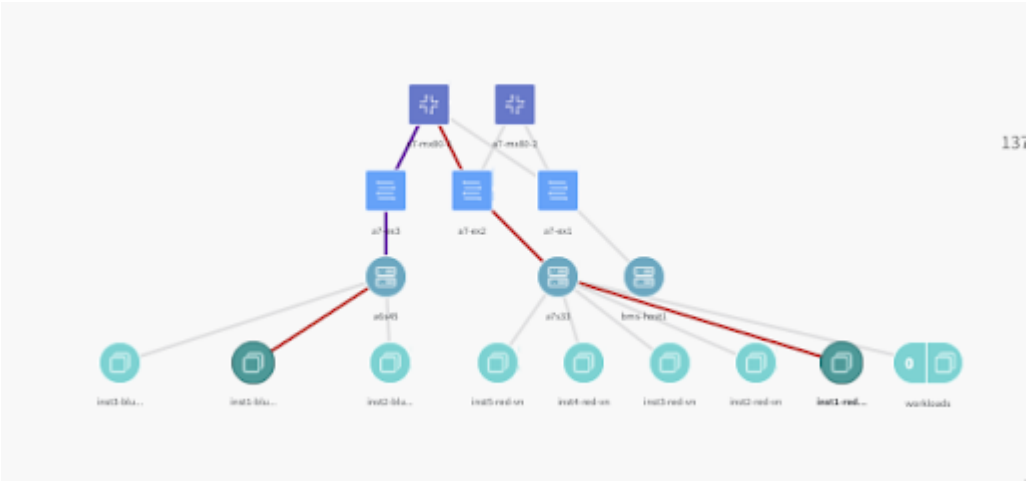
1. Log in to a cluster using Contrail Command by browsing to:

   https://<*Contrail-Command-Server-IP-Address*>:9091

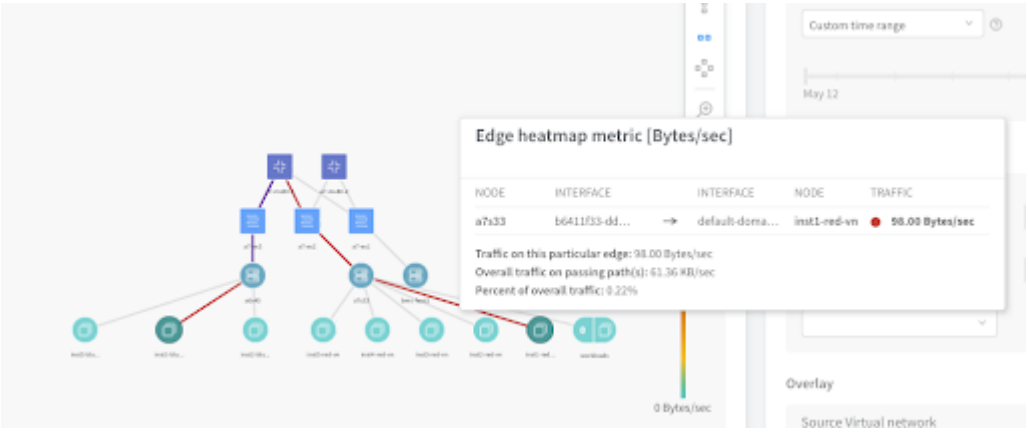
2. Select **Infrastructure > Fabrics**.
3. Select the desired **Fabric** from the available list.
4. Click **Topology View**.

The packet path will look similar to the following example.

Figure 132: Packet Path in Topology View

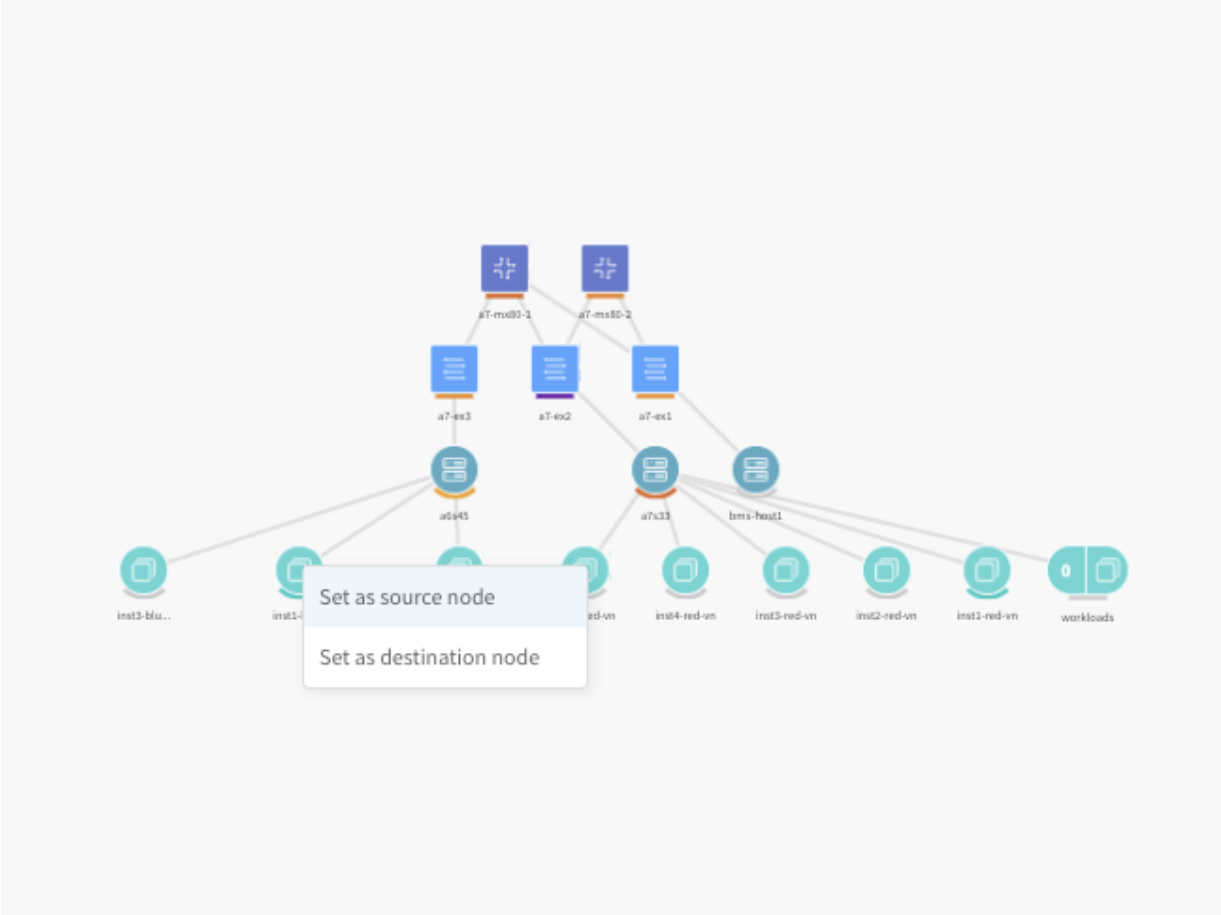


Right-click to show details about traffic on the path.



To change a path or create a new one, right-click to select the packet path source and destination nodes.

Figure 133: Packet Path Source and Destination nodes



The following options are available to filter the path.

Table 50: Packet Path Filter Options

Packet Path Option	Description
Predefined Time	Select the period in the history for which data is to be displayed.
Time Range	Use the calendar or type directly into the fields to select the desired start and end time. Additionally, you can select a time interval by moving the selector left or right..
Underlay	
Source IP	Filter data with the source IP address.
Source Port	Filter data with the source port.
Destination IP	Filter data with the destination IP address.
Destination Port	Filter data with the destination port.
Protocol	Filter data with these available protocol types: • ICMP • OSPFIGP • TCP • UDP
Overlay	
Source Virtual Network	Filter data with the source virtual network.
Source IP	Filter data with the source IP address.
Source Port	Filter data with the source port.

Table 50: Packet Path Filter Options *(Continued)*

Packet Path Option	Description
Destination Virtual Network	Filter data with the destination virtual network.
Destination IP	Filter data with the destination IP address.
Destination Port	Filter data with the destination port.
Protocol	Filter data with these available protocol types: • ICMP • TCP • UDP

Fabric-Only Path Finding

Contrail Networking Release 2011 supports fabric-only path finding.

In fabric-only path finding, each virtual port group (VPG) is treated as a bare metal server (BMS). These BMSs are named `bms_<vpg_name>`. Contrail Networking does not allocate IP addresses of the VPG. The IP addresses are discovered by mapping the Contrail Config `virtual-port-group` object with the gRPC (gRPC remote procedure calls) Sensor data, `"/network-instances/network-instance/macip-table"`.

When a Contrail vRouter Agent is present in the path of a flow between two virtual machines that is hosted on Contrail compute nodes, Contrail Insights Flows collector receives information from Contrail Flows for the corresponding sFlow packet. However, when the traffic flow is between two bare metal servers and there is no Contrail vRouter Agent, no information is received from Contrail flows. In this scenario, the gRPC sensor is subscribed to, to determine virtual network information and connections between leaf and BMS.

NOTE: Fabric-only path finding feature is supported on network devices that run Junos OS Release 20.2 or later.

Change History Table

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

Release	Description
2011	Contrail Networking Release 2011 supports fabric-only path finding.
2008	Starting in Contrail Networking Release 2008, you can view the path a packet takes in a network.

RELATED DOCUMENTATION

Viewing Topology Maps from Contrail Command | 286

Contrail Insights Flows in Contrail Command | 339

Assign Custom Names to Privileged Ports and VXLAN IDs

IN THIS SECTION

- Assign a Custom Name to Privileged Ports | 296
- Map a Route with Custom Named Ports | 299
- Search for a Port Using the Custom Name | 300
- Assign a Custom Name to VXLAN IDs | 300

Starting with Contrail Networking Release 2011, you can assign custom names to privileged ports (for example HTTP, HTTPs, BGP, DNS, SSH) and VXLANs in order to make them easier to identify in the topology mapping and queries. When you make queries for traffic flows in the Topology View, the custom name displays instead of the port number or VXLAN ID.

Assign a Custom Name to Privileged Ports

To assign a custom name to a privileged port:

1. Navigate to **Infrastructure** > **Fabrics**.

   The Fabrics page displays.

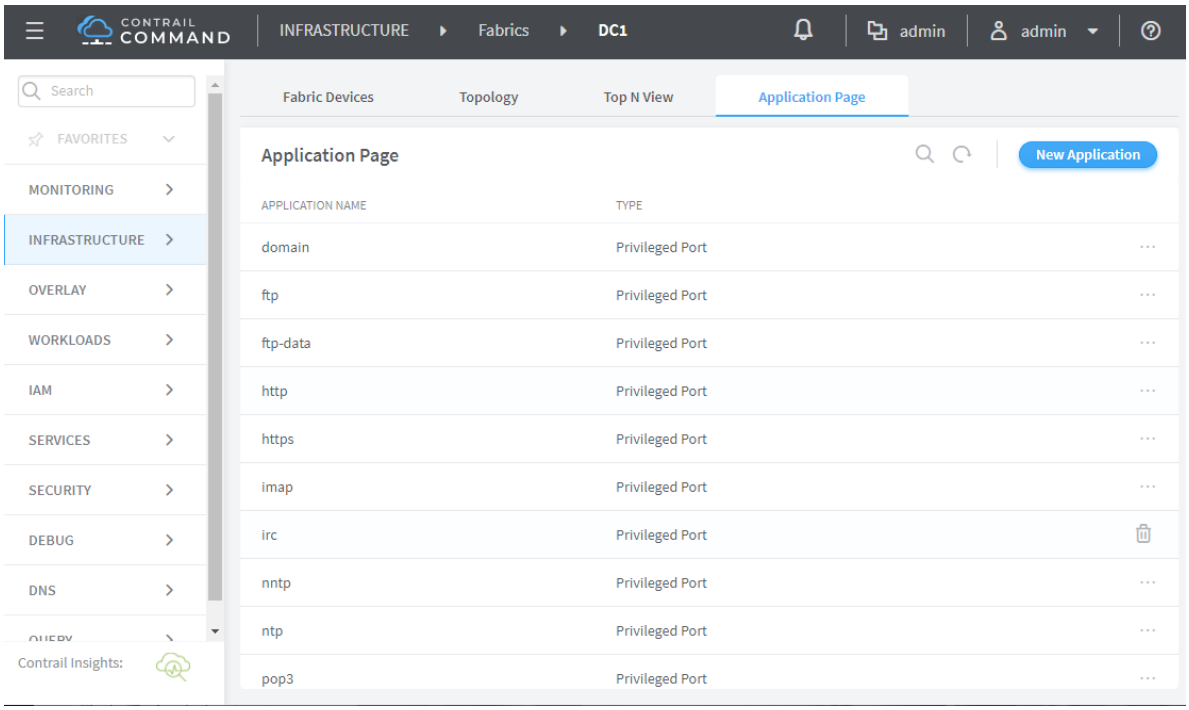
2. Select *<Fabric Name>*.

   The Fabric Devices page displays for the selected fabric.

3. Click the **Application Page** tab.

   By default, all the privileged ports are displayed.

Figure 134: Application Page Showing Privileged Ports



4. Click **New Application**.

   The Create new application dialog box opens.

Figure 135: Create New Application

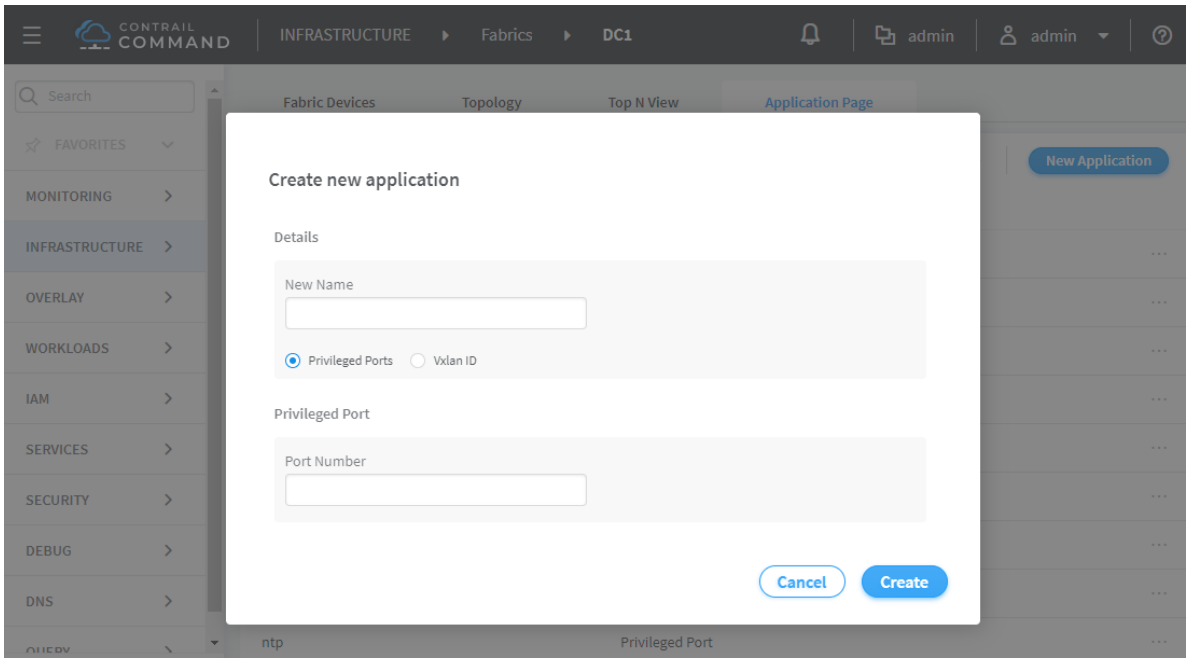


Table 51: Create New Application: Port Field Information

Field	Description
New Name	Enter the new custom name.
Privileged Ports	Select to assign the new custom name to a privileged port.
Port Number	Enter a port number to rename with the new custom name.

5. Click **Create**.

   You are returned to the Application Page.

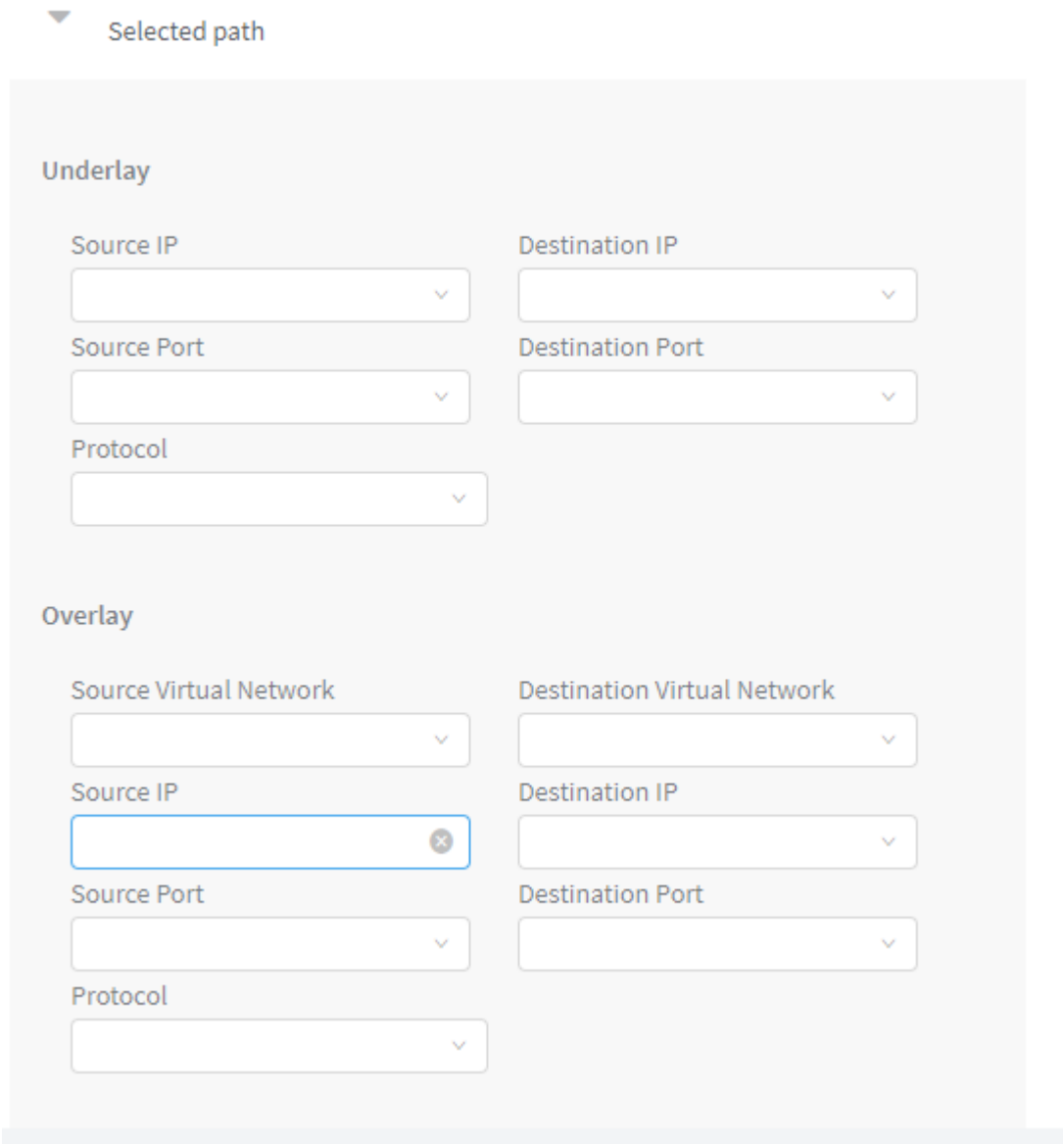
6. Click the **Topology** tab.

   The topology for the fabric displays.

7. In the Topology map, hover over the node, and select **Set as Source node**.

8. In the Overlay section in the topology sidebar, select that same **Source IP** from the drop-down list.

Figure 136: Overlay Source IP



Locate the port number that you named, which shows the custom name.

Map a Route with Custom Named Ports

Custom names were previously assigned to two ports and the custom names are shown for two host IP addresses.

To map a route with custom named ports for both the source IP address and destination IP address:

1. Navigate to **Infrastructure** > **Fabrics**.

The Fabrics page displays.

2. Select *<Fabric Name>*.

   The Fabric Devices page displays for the selected fabric.

3. Click the **Topology** tab.

   The topology for the fabric displays.

4. In the Overlay section in the topology sidebar, select **Source IP** from the drop-down list.

5. Select a port with a custom name next to the port number.

6. Select **Destination IP** from the drop-down list.

7. Select a port with a custom name next to the port number.

   A path is shown between the two selected nodes.

Search for a Port Using the Custom Name

A custom name was previously assigned to a port.

To search for a port using custom name:

1. Navigate to **Infrastructure** > **Fabrics**.

   The Fabrics page displays.

2. Select *<Fabric Name>*.

   The Fabric Devices page displays for the selected fabric.

3. Click the **Topology** tab.

   The topology for the fabric displays.

4. In the Overlay section in the topology sidebar, select **Source IP** from the drop-down list.

5. Enter the custom name in the **Port** field.

   Port drop-down shows only the port with the custom name.

Assign a Custom Name to VXLAN IDs

To assign a custom name to VXLAN ID:

1. Navigate to **Infrastructure** > **Fabrics**.

The Fabrics page displays.

2. Select *<Fabric Name>*.

   The Fabric Devices page displays for the selected fabric.

3. Click the **Application Page** tab.

   By default, all the privileged ports are displayed.

4. Click **New Application**.

   The Create new application dialog box opens.

   Table 52: Create New Application: VXLAN Field Information

Field	Description
New Name	Enter the new custom name.
VXLAN ID	Select to assign the new custom name to a VXLAN ID of a corresponding Virtual Network.
VXLAN ID	Enter a VXLAN ID to rename with the new custom name.

5. Click **Create**.

   You are returned to the Application page.

6. Click the **Topology** tab.

   The topology for the fabric displays.

7. In the Topology map, hover over the node, and select **Set as Source node**.

8. In the Overlay section in the topology sidebar, select **Source Virtual Network** from the drop-down list.

9. Locate the VXLAN ID of a Virtual Network that you named.

   Next to the VXLAN ID in the drop-down list is the custom name you added.

Change History Table

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

Release	Description
2011	Starting with Contrail Networking Release 2011, you can assign custom names to privileged ports (for example HTTP, HTTPs, BGP, DNS, SSH) and VXLANs in order to make them easier to identify in the topology mapping and queries. When you make queries for traffic flows in the Topology View, the custom name displays instead of the port number or VXLAN ID.

RELATED DOCUMENTATION

- Top N View in Contrail Command | 280
- Viewing Topology Maps from Contrail Command | 286
- Viewing Packet Path in Topology View | 291

Viewing the Monitoring Dashboards

IN THIS SECTION

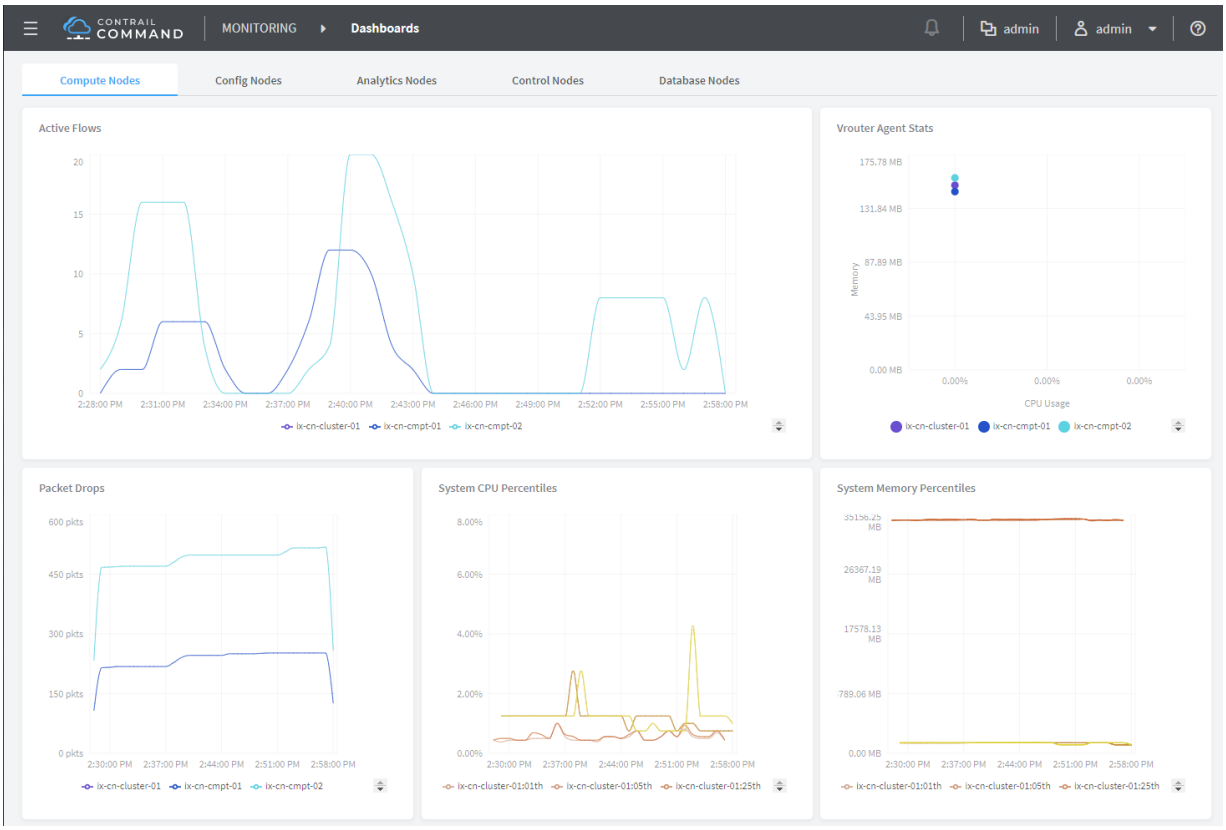
- Monitoring Dashboards | 302
- Monitor Individual Details from the Dashboard | 303
- Chart Data Values | 303
- Monitor Config Nodes | 303
- Monitor Analytics Nodes | 304

The **Dashboards** page in Contrail Command provides an “at-a-glance” view of the system infrastructure components, including the number of compute nodes, config nodes, analytics nodes, control nodes, and database nodes currently operational, CPU and memory utilization for the system, active flows, and packet drops.

Monitoring Dashboards

Select **Monitoring > Dashboards** to view the **Dashboards** page.

Figure 137: Monitoring > Dashboards



Monitor Individual Details from the Dashboard

Across the top of the **Dashboards** page are tabs representing the components of the system that are shown in the statistics. See Figure 137 on page 303 . Any of the compute nodes, config nodes, analytics nodes, control nodes, and database nodes can be monitored individually and in detail from the **Dashboards** page by clicking an associated tab, and drilling down for more detail.

Chart Data Values

The charts show the latest data, updating in real-time from a stream of data from the Contrail 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 using the mouse scroll wheel.

Monitor Config Nodes

Select **Monitoring > Dashboards > Config Nodes** to view a summary of activities for the analytics nodes. See Figure 138 on page 304 . Hover over any chart axis to get summary information about the component it represents.

Figure 138: Config Nodes Summary

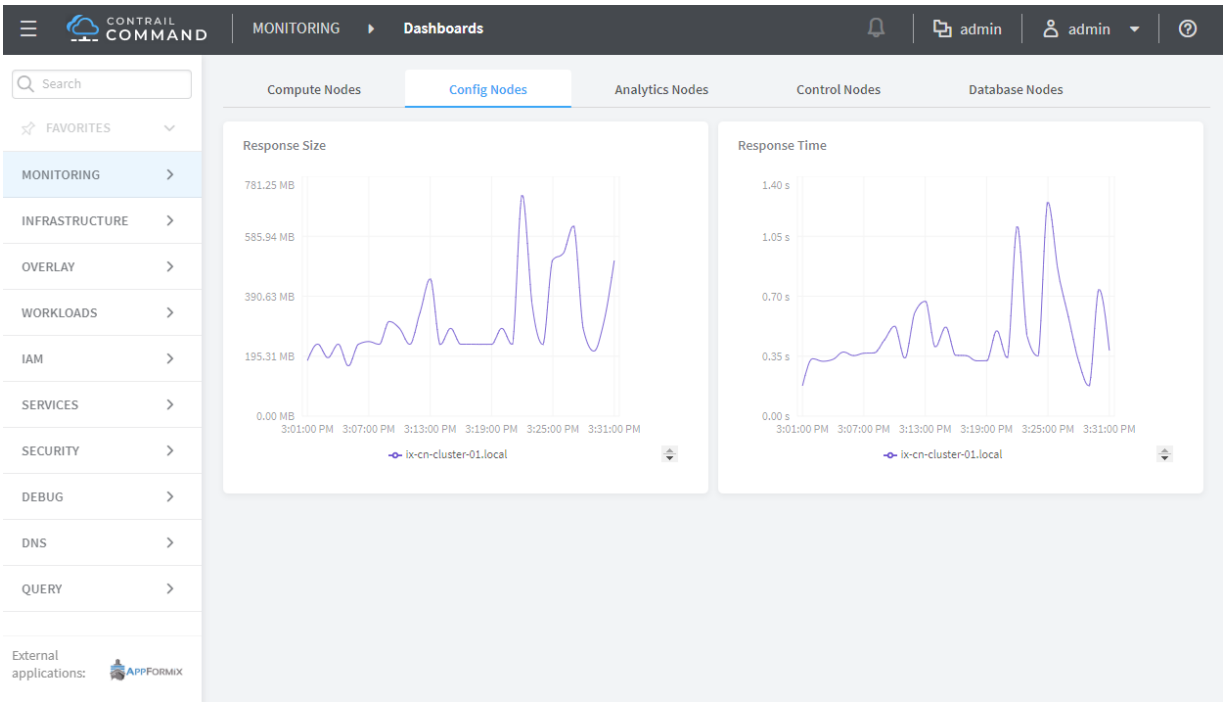


Table 53 on page 304 describes the fields in the Config Nodes summary.

Table 53: Config Nodes Summary Fields

Field	Description
Host Name	The name of this node.
Response Size	Response size (MB) at specified time.
Response Time	Response time (ms) at specified time.

Monitor Analytics Nodes

Select **Monitoring > Dashboards > Analytics Nodes** to view a summary of activities for the analytics nodes. See Figure 139 on page 305 . Hover over any chart axis or bubble to get summary information about the component it represents. See Table 54 on page 305 for descriptions of the fields on the analytics summary.

Figure 139: Analytics Node Summary

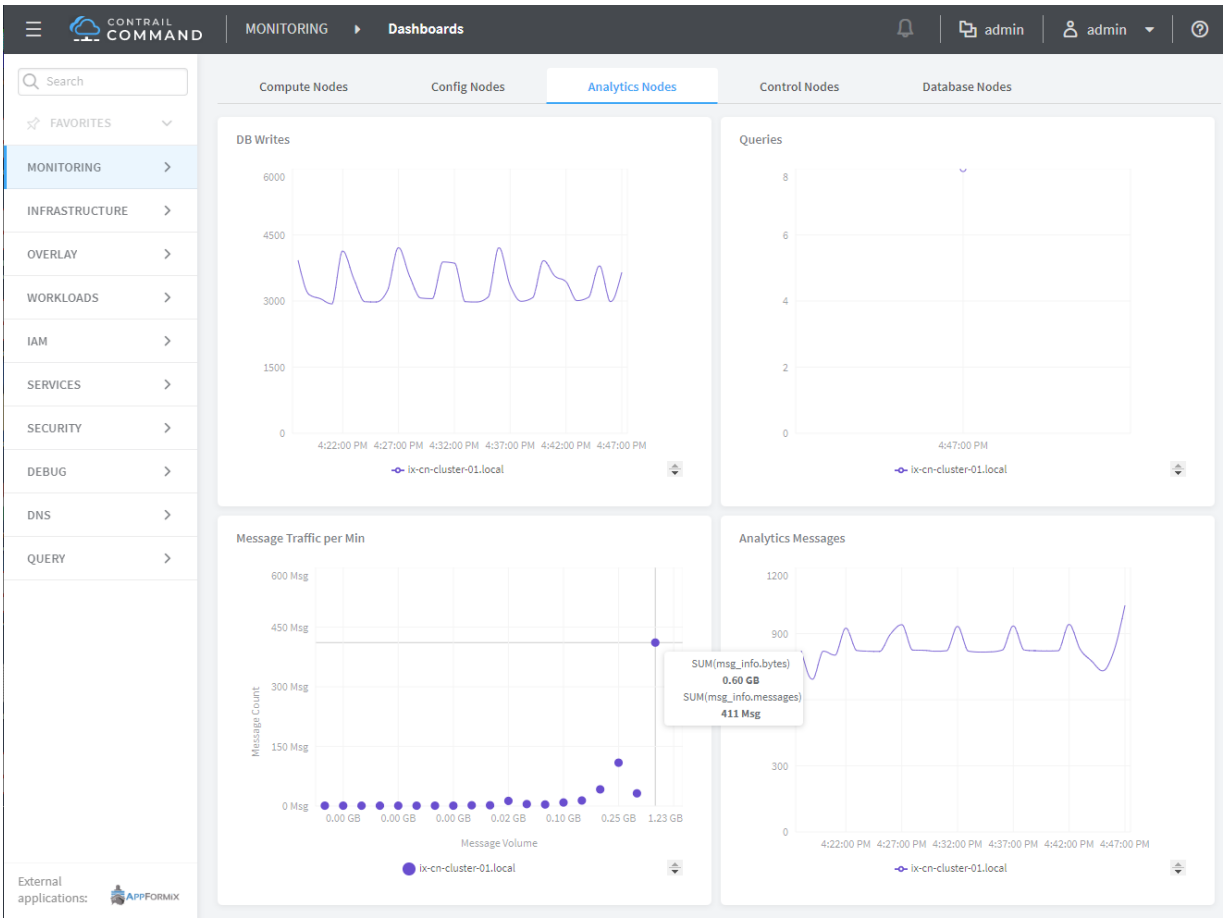


Table 54: Fields in Analytics Nodes Summary

Field	Description
Host Name	The name of this node.
DB Writes	Number of writes to database in Contrail made in specific time period.
Message Traffic per Min	Number of messages received per minute.
Queries	Number of queries made and time run.

Table 54: Fields in Analytics Nodes Summary *(Continued)*

Field	Description
Analytics Messages	Number of analytics messages received during specific time period on represented node.

Creating a Query for Flows

IN THIS SECTION

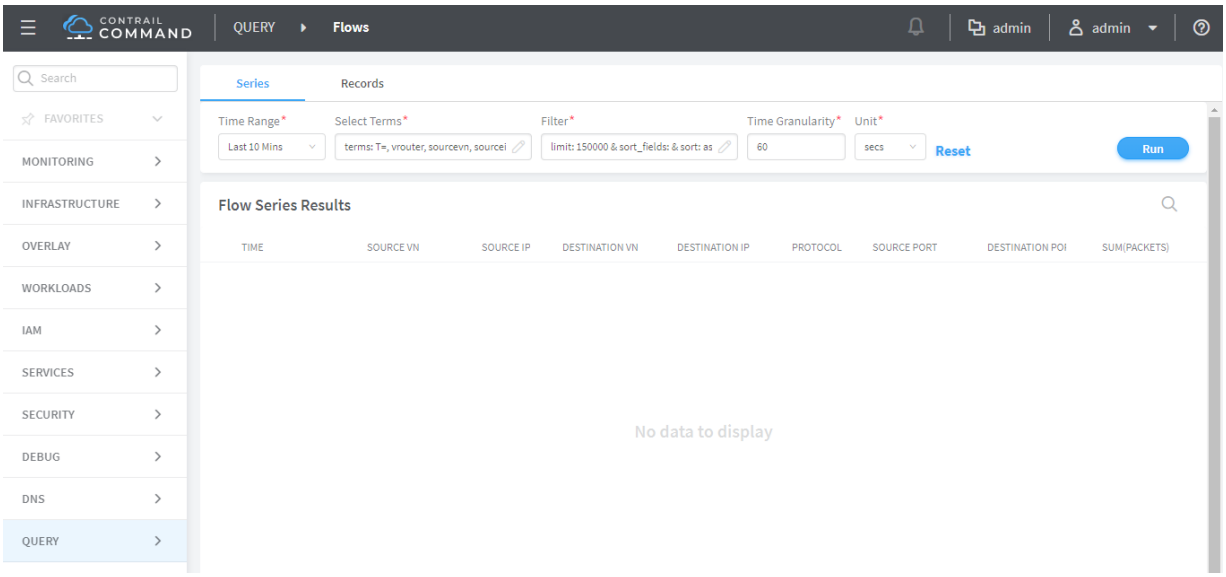
- Query Flow Series Table | **306**
- Query Individual Flow Records | **310**
- Using the Query Window | **313**
- Display Flows Query Queue | **314**

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 Flow Series Table

Select **Query > Flows > Series** to create queries of the flow series table. The results are in the form of time series data for flow series. See Figure 140 on page 307 .

Figure 140: Query Flow Series Window



The query fields available on the screen for the **Series** tab are described in Table 55 on page 307 . Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 55: Query Flow Series Fields

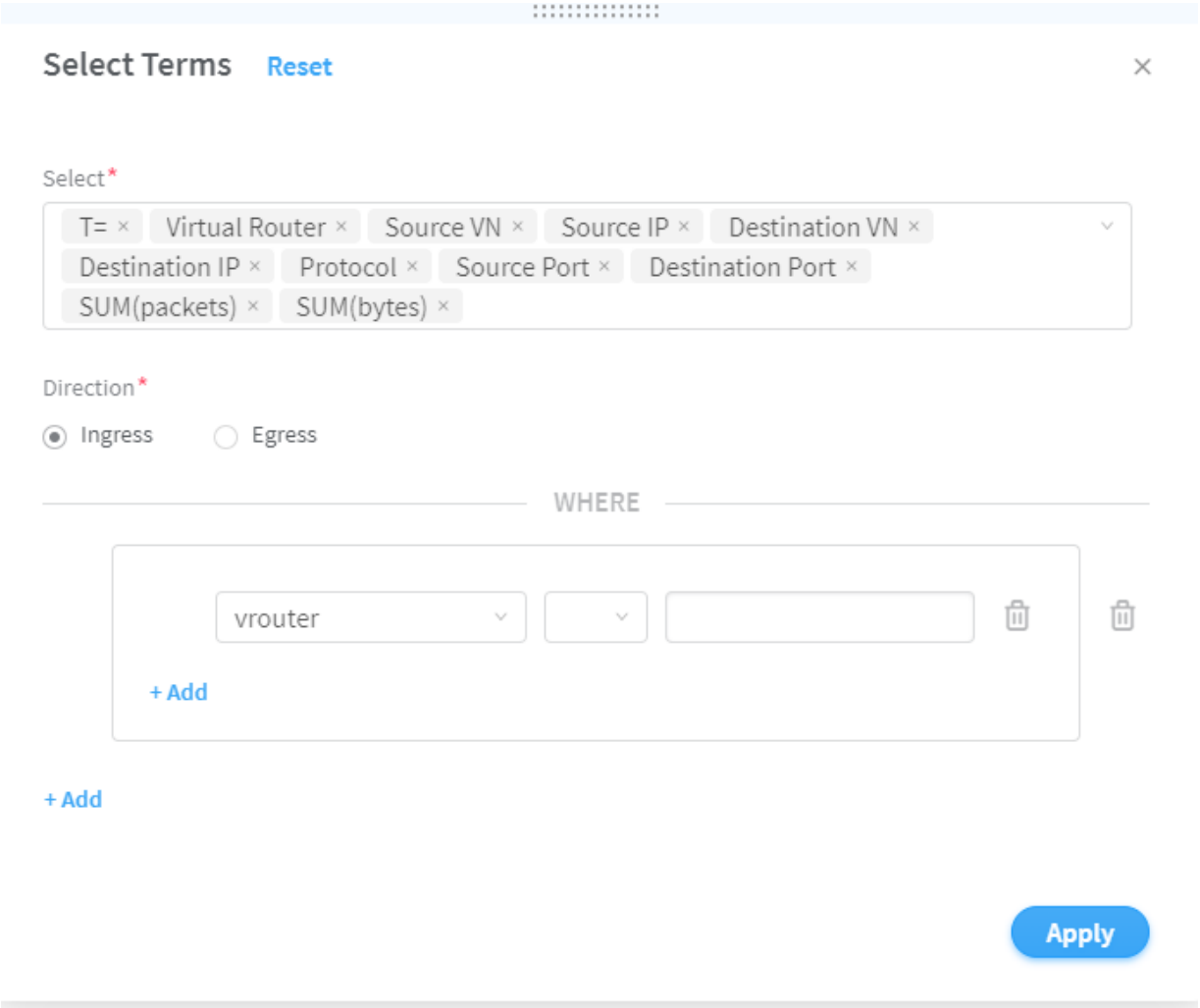
Field	Description
Time Range	Select a range of time to display the flow series: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specific custom time range in two fields: **Start Time** and **End Time**.
Select Terms	Click the edit button (pencil icon) to open a **Select Terms** window (Figure 141 on page 309), where you can click one or more fields to display from the flow series, such as **Virtual Router, Source VN, Destination VN, SUM(bytes), SUM(packets)**, and more.

Table 55: Query Flow Series Fields *(Continued)*

Field	Description
Direction	Select the desired flow direction: **Ingress** or **Egress**.
Where	Click the **+Add** to open a query-writing window, where you can specify query values for variables such as **destvn, protocol, sourcevn, and vrouter**.
Filter	Click the edit button (pencil icon) to open a **Filters** window (Figure 140 on page 307), where you can select filter items to sort by, the sort order, and limits to the number of results returned.
Time Granularity	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.
Unit	Select minutes or seconds for unit of measurement.
Run	Click **Run** 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.
Export	The Export button is displayed after you click **Run**. This allows you to export the list of flows to a text .csv file.

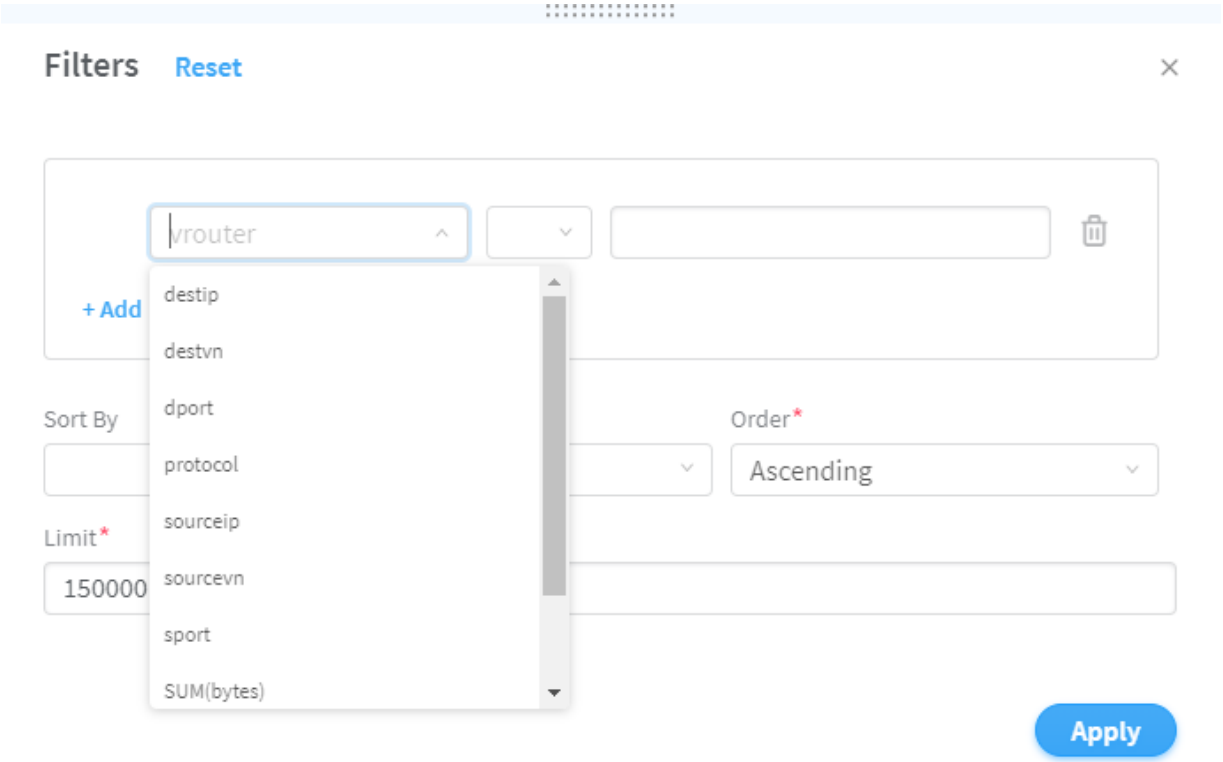
The **Select Terms** window allows you to select one or more attributes of a flow series by clicking each attribute desired. See Figure 141 on page 309 . Select **SUM(Bytes)** or **SUM(Packets)** to aggregate bytes and packets in intervals.

Figure 141: Flow Series Select Terms



Use the **Filters** 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 142 on page 310 .

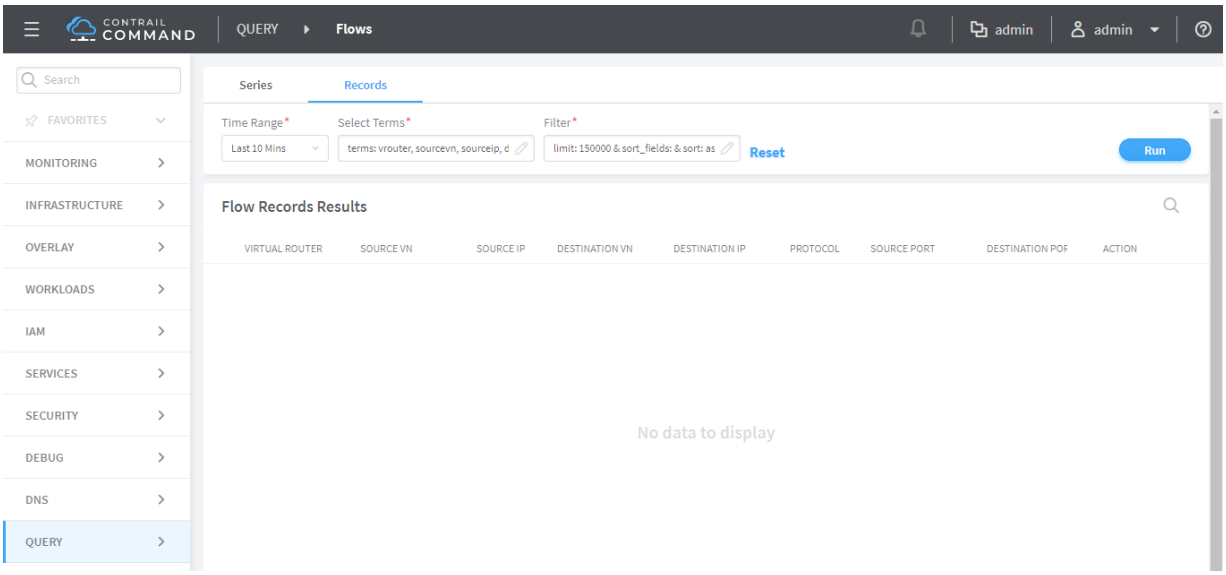
Figure 142: Flows Series Filter



Query Individual 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 143: Flows Records



The query fields available on the screen for the **Records** tab are described in Table 56 on page 311 . Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 56: Query Flow Records Fields

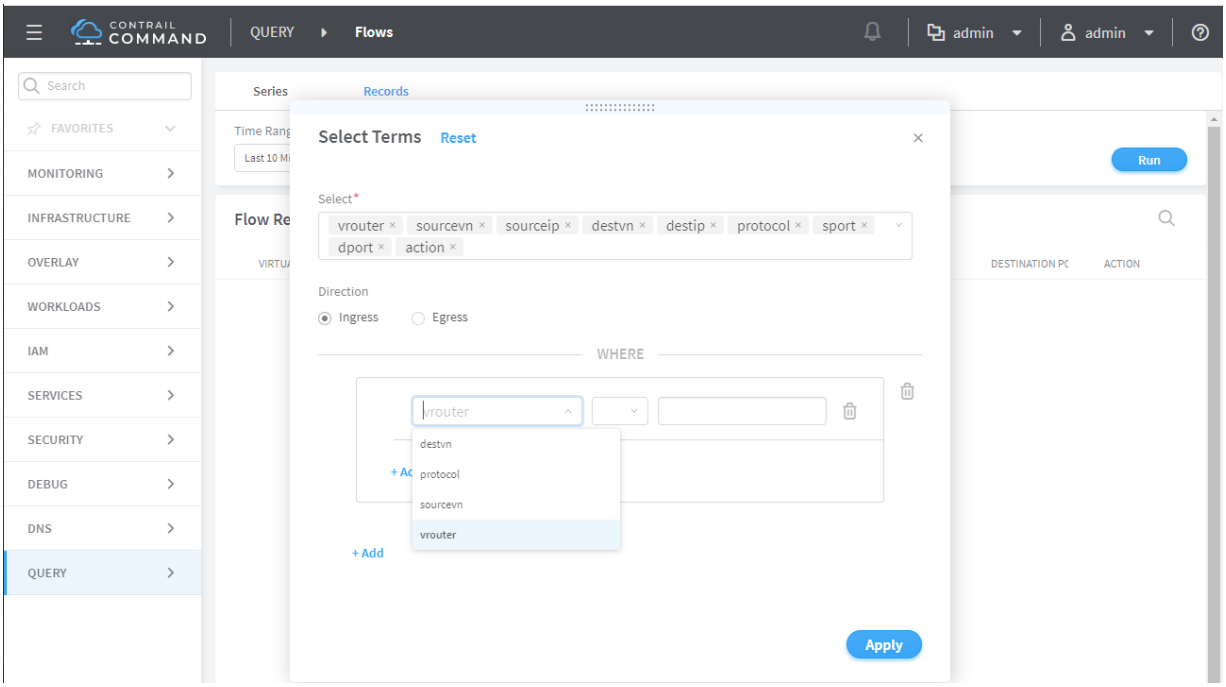
Field	Description
Time Range	Select a range of time for the flow records: • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom Click **Custom** to enter a specified custom time range in two fields: **Start Time** and **End Time**.
Select Terms	Click the edit button (pencil icon) to open a **Select Terms** window (Figure 143 on page 311), where you can click one or more attributes to display for the flow records, including **vrouter, sourcevn, sourceip, destvn, destip, protocol, dport,** and **action**.

Table 56: Query Flow Records Fields *(Continued)*

Field	Description
Direction	Select the desired flow direction: **Ingress** or **Egress**.
Where	Click **+Add** to open a query window where you can specify query values for **destvn, protocol, sourcevn,** and **vrouter**.
Run	Click **Run** 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**, allowing you to export the list of flows to a text **.csv** file.

The **Select Terms** window allows you to select one or more attributes to display for the flow records selected. See Figure 144 on page 312 .

Figure 144: Flows Records Select Terms



Using the Query Window

The query window is available by clicking the **+Add** in the **Where** field. Use the query window to enter query statements. See Figure 144 on page 312 .

You can restrict the query to a particular source VN and destination VN combination using the **Select** section.

The **Where** 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 **sourcevn = vn1**.

See Table 57 on page 313 for descriptions of the fields in the query window.

Table 57: Query Window Fields and Descriptions

Item	Description
—	Select from a list of available item types the type from which to query. • destvn • protocol • sourcevn • vrouter
(operator)	**=**(equal to) and **Starts with** are available.
AND +	Click the **+Add** to add more elements to your query. Repeat to include additional query elements to your query statement.
Apply	Click to enter the query into the fields on the main screen.

The **Where** clause supports logical AND and logical OR operations.

The **Where** can be modeled as logical OR of multiple AND terms. ((term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1.**

Display Flows Query Queue

Select **Query > Flows > Query Queue** to display queries that are in the queue waiting to be performed on the data.

The query fields available on the screen for the **Records** tab are described in Table 58 on page 314 . Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 58: 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

Fat Flows

Contrail Analytics Optional Modules

IN THIS SECTION

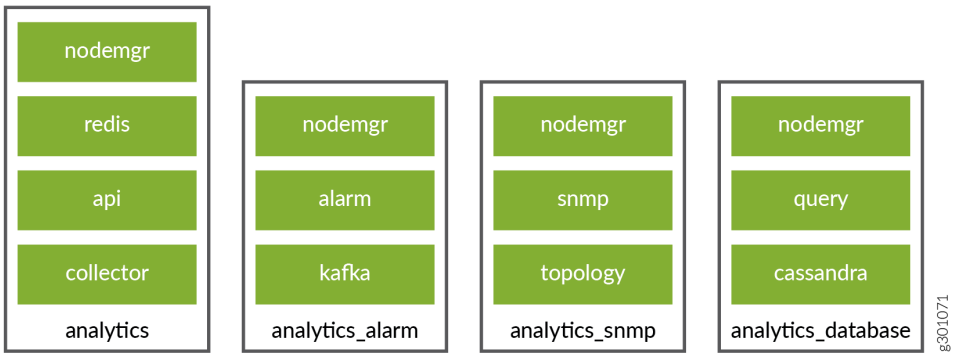
- Analytics Optional Components | 315
- Contrail Web UI | 324
- Tripleo Provisioning | 331
- Appendix | 332

Analytics Optional Components

Contrail analytics is comprised of four building blocks. The last three listed are optional components.

- Analytics collector
- Analytics alarm
- Analytics SNMP
- Analytics database

Figure 145: Contrail Analytics Components



Regardless that the alarm, SNMP, and database analytics roles have not been installed and if installed are disabled, these components show as active when you run the Linux $sudo commands or view in Contrail Command. For more information, see the section “TripleO Provisioning” below.

Contrail Infrastructure Installed without Optional Analytics Components

Two topologies are considered in this example: multi-nodes or single node.

Multi-nodes Contrail controller components are split onto three servers (Contrail controller, Contrail analytics, and Contrail analytics database). Only the first two servers are mandatory since Contrail analytics database is an optional component. This type of topology is used in production deployments.

Single node This type of topology is used in test deployments.

Multi-Nodes Contrail Controller

Contrail Controller Node

Following is an example of the Contrail status on the Contrail controller node:

```
$ sudo contrail-status
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active

== Contrail webui ==
web: active
job: active
```

Contrail Analytics Node (with All Optional Components)

Following is the Contrail status on Contrail analytics node when Contrail analytics SNMP and Contrail analytics alarm have both been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
analytics api contrail-analytics-api rhel-
queens-1910-23 running 62980f3e6479 Up 2 weeks
analytics collector contrail-analytics-collector rhel-
queens-1910-23 running b777437946c2 Up 2 weeks
analytics nodemgr contrail-nodemgr rhel-
queens-1910-23 running aeeb744a5b5e Up 2 weeks
analytics redis contrail-external-redis rhel-
queens-1910-23 running 150b6225bd93 Up 2 weeks
analytics-alarm alarm-gen contrail-analytics-alarm-gen rhel-
queens-1910-23 running d655146cb8d0 Up 2 weeks
analytics-alarm kafka contrail-external-kafka rhel-
queens-1910-23 running 8cfa8c7da4bd Up 2 weeks
analytics-alarm nodemgr contrail-nodemgr rhel-
queens-1910-23 running 685a5f817f0b Up 2 weeks
analytics-alarm zookeeper contrail-external-zookeeper rhel-
queens-1910-23 running a41dc5658c72 Up 2 weeks
analytics-snmp nodemgr contrail-nodemgr rhel-
queens-1910-23 running 0afd301ccbd8 Up 2 weeks
analytics-snmp snmp-collector contrail-analytics-snmp-collector rhel-
queens-1910-23 running 2bde6aa39250 Up 2 weeks
analytics-snmp topology contrail-analytics-snmp-topology rhel-
queens-1910-23 running a16f983ed162 Up 2 weeks

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail analytics-snmp ==
```

```
snmp-collector: active
nodemgr: active
topology: active
```

Contrail analytics alarm and SNMP are deployed and active.

Contrail Analytics Node (without Analytics Optional Components)

Following is an example of the Contrail status on Contrail analytics node when Contrail analytics SNMP and Contrail analytics alarm have not been deployed:

```
$ sudo contrail-status
Pod Service Original Name Original Version
State Id Status
analytics api contrail-analytics-api rhel-queens-2005-62
running 489b07cbbbef Up 18 hours
analytics collector contrail-analytics-collector rhel-queens-2005-62
running 5da4f99b045f Up 18 hours
analytics nodemgr contrail-nodemgr rhel-queens-2005-62
running 28053f64f1bc Up 18 hours
analytics provisioner contrail-provisioner rhel-queens-2005-62
running faa8de6d17e4 Up 18 hours
analytics redis contrail-external-redis rhel-queens-2005-62
running 3e29dcc475d1 Up 18 hours
analytics stunnel contrail-external-stunnel rhel-queens-2005-62
running 11a30f0f5e3b Up 18 hours

== Contrail analytics ==
nodemgr: active
api: active
collector: active
```

Only Contrail analytics collector is deployed and active.

Contrail Analytics Database Node

Contrail analytics database is only deployed when the analytics database component is enabled. The following example shows the Contrail status on the Contrail analytics database node:

```
$ sudo contrail-status
Pod Service Original Name Original Version
```

```
State Id Status
database cassandra contrail-external-cassandra rhel-queens-1910-
23 running ec05bd8c34c4 Up 2 weeks
database nodemgr contrail-nodemgr rhel-queens-1910-
23 running 25a6c58d5144 Up 2 weeks
database query-engine contrail-analytics-query-engine rhel-queens-1910-
23 running f90f7ae16b48 Up 2 weeks

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active
```

Single Node Contrail Controller

Contrail Controller Node (with All Analytics Optional Components)

Following is the Contrail status on Contrail controller node when Contrail analytics SNMP, Contrail analytics alarm, and Contrail analytics database have been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
analytics api contrail-analytics-api
rhel-queens-1912-46 running bf87cc51fb36 Up 8 weeks
analytics collector contrail-analytics-collector
rhel-queens-1912-46 running 0ae1ca0fb1f2 Up 8 weeks
analytics nodemgr contrail-nodemgr
rhel-queens-1912-46 running 24e9174056d0 Up 8 weeks
analytics redis contrail-external-redis
rhel-queens-1912-46 running 9d7135b6b9d8 Up 8 weeks
analytics stunnel contrail-external-stunnel
rhel-queens-1912-46 running 30d413bad4f1 Up 8 weeks
analytics-alarm alarm-gen contrail-analytics-alarm-gen
rhel-queens-1912-46 running 2f40aeb42154 Up 8 weeks
analytics-alarm kafka contrail-external-kafka
rhel-queens-1912-46 running 8cd54b9520af Up 8 weeks
analytics-alarm nodemgr contrail-nodemgr
rhel-queens-1912-46 running afeadd231273 Up 8 weeks
analytics-alarm zookeeper contrail-external-zookeeper
rhel-queens-1912-46 running 118b116b2721 Up 8 weeks
```

```
analytics-snmp nodemgr contrail-nodemgr
rhel-queens-1912-46 running f623346fff53 Up 8 weeks
analytics-snmp snmp-collector contrail-analytics-snmp-collector
rhel-queens-1912-46 running 152b037af72d Up 8 weeks
analytics-snmp topology contrail-analytics-snmp-topology
rhel-queens-1912-46 running 5226847e74f3 Up 8 weeks
config api contrail-controller-config-api
rhel-queens-1912-46 running b8ba22697cfe Up 8 weeks
config device-manager contrail-controller-config-devicemgr
rhel-queens-1912-46 running 29f9b248f850 Up 8 weeks
config nodemgr contrail-nodemgr
rhel-queens-1912-46 running 2f3f84d5d2b4 Up 8 weeks
config schema contrail-controller-config-schema
rhel-queens-1912-46 running 334906b962fb Up 8 weeks
config svc-monitor contrail-controller-config-svcmonitor
rhel-queens-1912-46 running a8581c37f9ab Up 8 weeks
config-database cassandra contrail-external-cassandra
rhel-queens-1912-46 running e47a3e430fe6 Up 8 weeks
config-database nodemgr contrail-nodemgr
rhel-queens-1912-46 running 4798399f0ec5 Up 8 weeks
config-database rabbitmq contrail-external-rabbitmq
rhel-queens-1912-46 running d80a5e8e8801 Up 8 weeks
config-database zookeeper contrail-external-zookeeper
rhel-queens-1912-46 running b1c430201497 Up 8 weeks
control control contrail-controller-control-control
rhel-queens-1912-46 running e478128385f7 Up 8 weeks
control dns contrail-controller-control-dns
rhel-queens-1912-46 running f9752a324d71 Up 8 weeks
control named contrail-controller-control-named
rhel-queens-1912-46 running 66c992adced5 Up 8 weeks
control nodemgr contrail-nodemgr
rhel-queens-1912-46 running 3c9a0270ab1a Up 8 weeks
database cassandra contrail-external-cassandra
rhel-queens-1912-46 running f85ead18fb26 Up 8 weeks
database nodemgr contrail-nodemgr
rhel-queens-1912-46 running 0d9f471003ea Up 8 weeks
database query-engine contrail-analytics-query-engine
rhel-queens-1912-46 running 40a092abbccf Up 8 weeks
webui job contrail-controller-webui-job
rhel-queens-1912-46 running 432f686a8abf Up 8 weeks
webui web contrail-controller-webui-web
rhel-queens-1912-46 running 4341432ce9a4 Up 8 weeks
```

```
== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics-alarm ==
nodemgr: active
kafka: active
alarm-gen: active

== Contrail database ==
nodemgr: active
query-engine: active
cassandra: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail webui ==
web: active
job: active

== Contrail analytics-snmp ==
snmp-collector: active
nodemgr: active
topology: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active
```

Contrail database (query), analytics alarm, and SNMP are deployed and active.

Contrail Controller Node (without Analytics Optional Components)

Following is an example of the Contrail status on Contrail controller node when Contrail analytics SNMP, Contrail analytics alarm, and Contrail analytics database have not been deployed:

```
$ sudo contrail-status
Pod Service Original Name
Original Version State Id Status
analytics api contrail-analytics-api
rhel-queens-2005-62 running b1ddca562595 Up 10 hours
analytics collector contrail-analytics-collector
rhel-queens-2005-62 running f6860911ee16 Up 10 hours
analytics nodemgr contrail-nodemgr
rhel-queens-2005-62 running 37a0d8744e31 Up 10 hours
analytics provisioner contrail-provisioner
rhel-queens-2005-62 running e2f9a4605d63 Up 10 hours
analytics redis contrail-external-redis
rhel-queens-2005-62 running 1d0a193983b0 Up 10 hours
analytics stunnel contrail-external-stunnel
rhel-queens-2005-62 running 695d61045e63 Up 10 hours
config api contrail-controller-config-api
rhel-queens-2005-62 running 41eb0caef12d Up 10 hours
config device-manager contrail-controller-config-devicemgr
rhel-queens-2005-62 running f3158c67d792 Up 10 hours
config nodemgr contrail-nodemgr
rhel-queens-2005-62 running 4138cc386e69 Up 10 hours
config provisioner contrail-provisioner
rhel-queens-2005-62 running 45aae86bb41a Up 10 hours
config schema contrail-controller-config-schema
rhel-queens-2005-62 running 2497392980d0 Up 10 hours
config svc-monitor contrail-controller-config-svcmonitor
rhel-queens-2005-62 running b2ed20209aa7 Up 10 hours
config-database cassandra contrail-external-cassandra
rhel-queens-2005-62 running abd3efad8075 Up 10 hours
config-database nodemgr contrail-nodemgr
rhel-queens-2005-62 running bcc74ecb37cc Up 10 hours
config-database provisioner contrail-provisioner
rhel-queens-2005-62 running 9de114119be5 Up 10 hours
config-database rabbitmq contrail-external-rabbitmq
rhel-queens-2005-62 running d623f5d3da79 Up 10 hours
```

```
config-database zookeeper contrail-external-zookeeper
rhel-queens-2005-62 running 2c4f47c2fdc1 Up 10 hours
control control contrail-controller-control-control
rhel-queens-2005-62 running 56e238791c60 Up 10 hours
control dns contrail-controller-control-dns
rhel-queens-2005-62 running 6cfc801451f9 Up 10 hours
control named contrail-controller-control-named
rhel-queens-2005-62 running f033a8bf5b88 Up 10 hours
control nodemgr contrail-nodemgr
rhel-queens-2005-62 running 7381053ff80f Up 10 hours
control provisioner contrail-provisioner
rhel-queens-2005-62 running a3851c25f427 Up 10 hours
webui job contrail-controller-webui-job
rhel-queens-2005-62 running 80cd5c06ff39 Up 10 hours
webui web contrail-controller-webui-web
rhel-queens-2005-62 running 51a2f164a259 Up 10 hours

== Contrail control ==
control: active
nodemgr: active
named: active
dns: active

== Contrail analytics ==
nodemgr: active
api: active
collector: active

== Contrail config-database ==
nodemgr: active
zookeeper: active
rabbitmq: active
cassandra: active

== Contrail config ==
svc-monitor: active
nodemgr: active
device-manager: active
api: active
schema: active

== Contrail webui ==
```

```
web: active
job: active
```

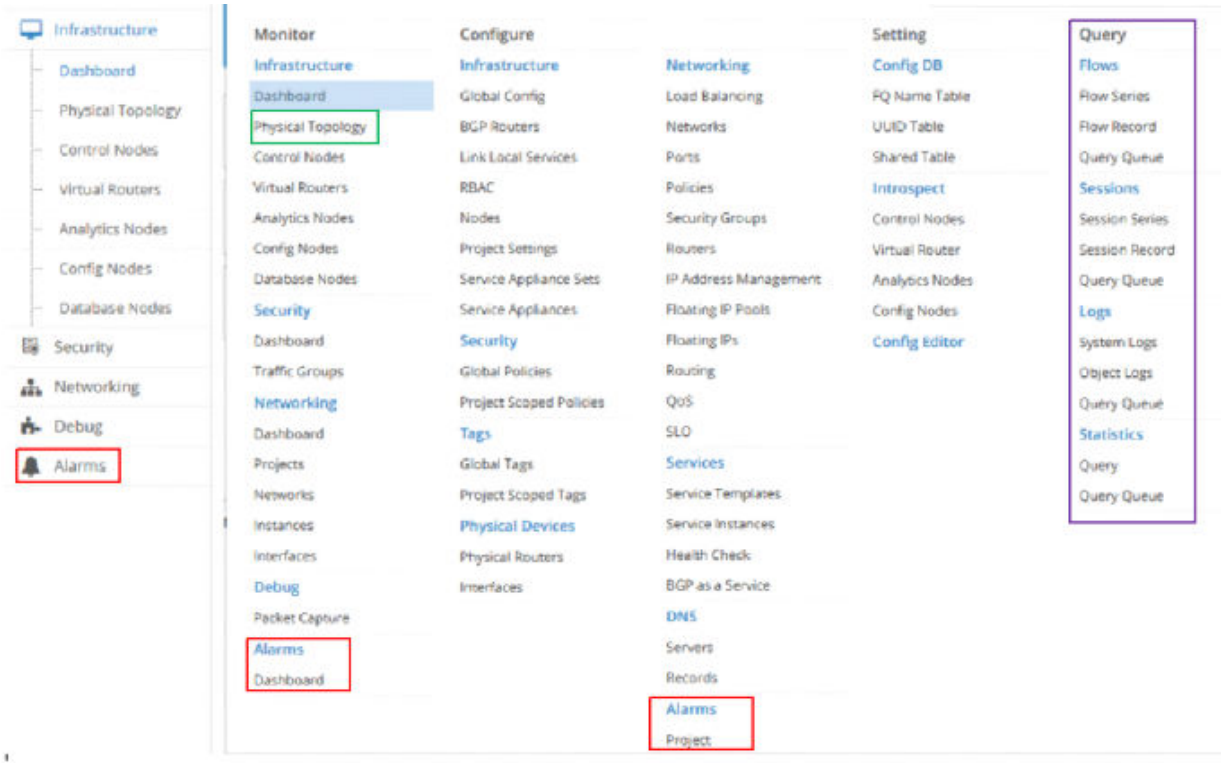
Contrail database (query), analytics alarm, and SNMP are not deployed.

Contrail Web UI

Web UI with Optional Components

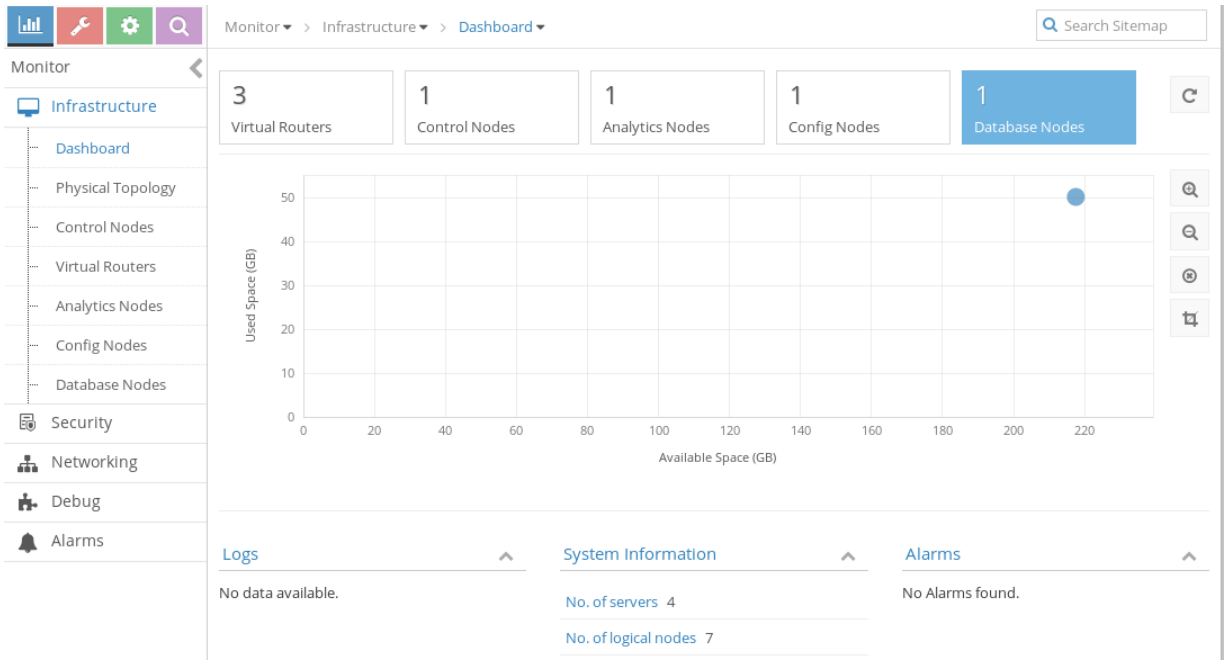
Figure 146 on page 324 displays the Contrail Web UI dashboard with all optional analytics components deployed.

Figure 146: Web UI - All Optional Analytics Components Deployed



A database node is visible in the infrastructure dashboard.

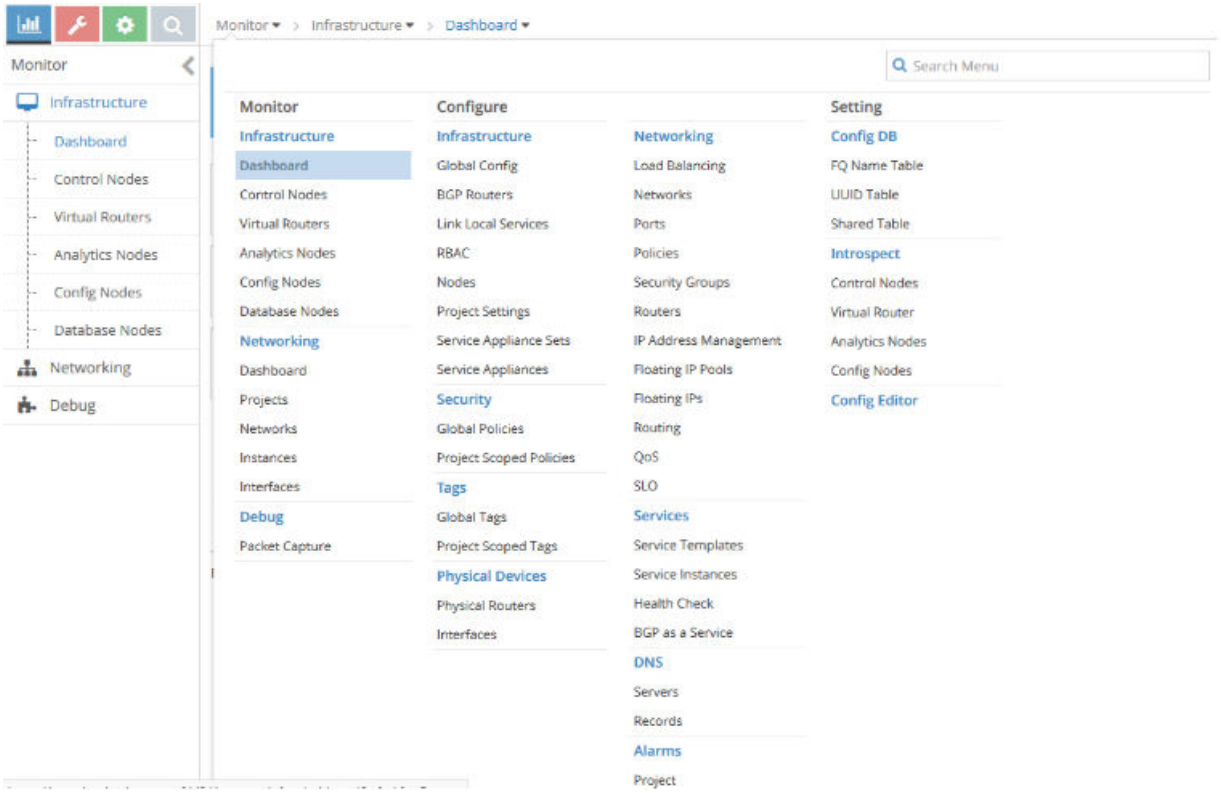
Figure 147: Web UI - Database Node in Dashboard



Web UI without Optional Components

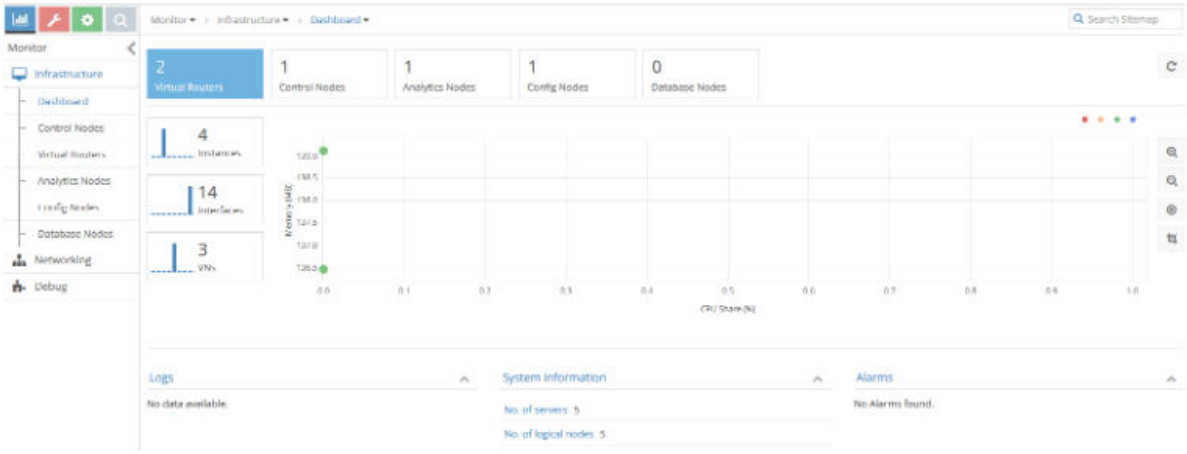
Figure 148 on page 326 displays the Contrail Web UI dashboard without optional analytics components deployed.

Figure 148: Web UI - Optional Analytics Components Not Deployed



No database node is visible in the infrastructure dashboard:

Figure 149: Web UI - Database Node Not Visible in Dashboard



Analytics Alarm Feature Enabled

Figure 150 on page 327 displays the **Monitor > Alarms** menu.

Figure 150: Web UI - Monitor > Alarms Menu

Figure 151 on page 327 displays the **Configure > Alarms** menu.

Figure 151: Web UI - Configure > Alarms Menu

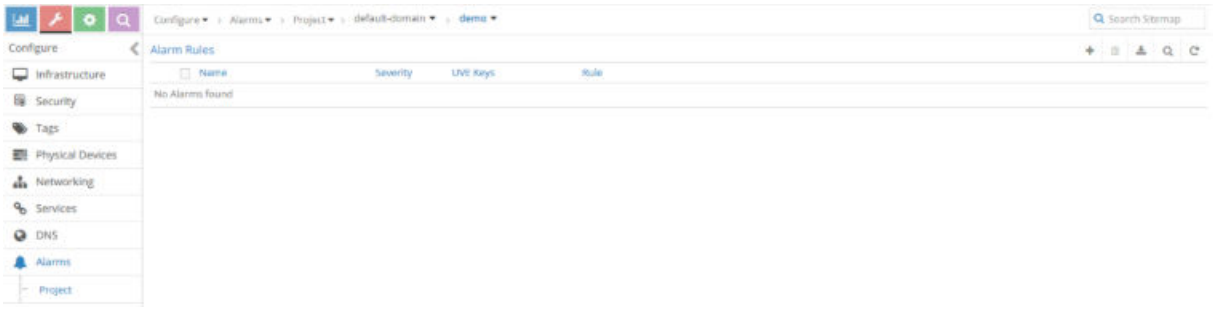
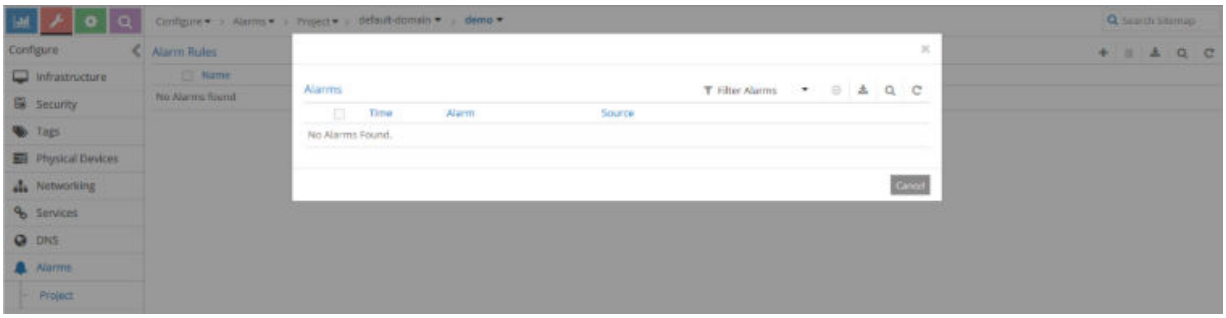


Figure 152 on page 328 displays the dialog box which appears when **Global Alarm**, next to Logged in User in the upper right, is selected.

Figure 152: Web UI - Global Alarm Settings



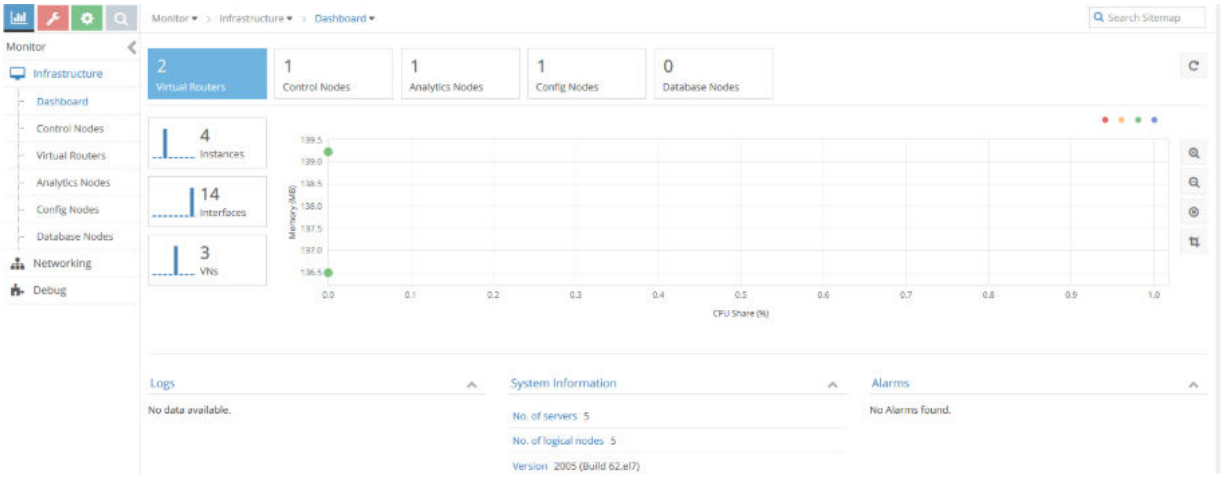
Analytics Alarm Feature Disabled

If the alarm analytics component is not deployed, then Contrail Web UI should not display the following alarm references:

- Global Alarm (Next to Logged in User)
- Monitor > Alarms
- Configure > Alarms

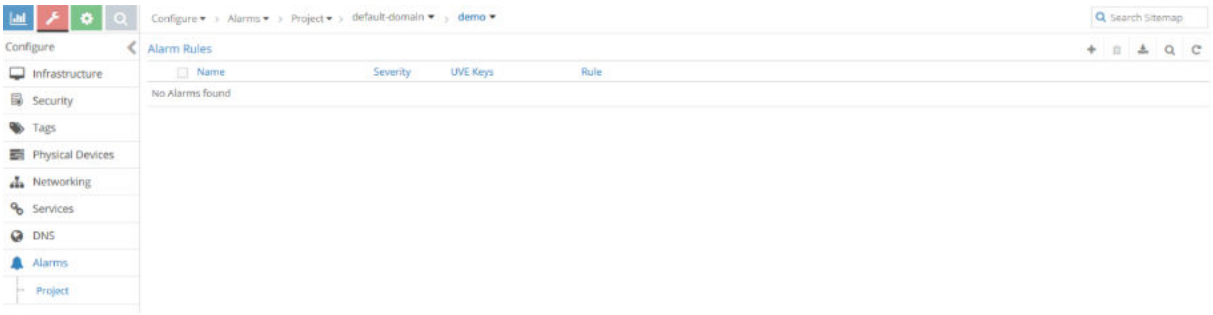
There is not an appearance of Global Alarm or **Alarms** entry in the Monitor menu:

Figure 153: Analytics Alarm Disabled - Global Alarm and Alarm Not Available



Alarms menu still available in Configure menu.

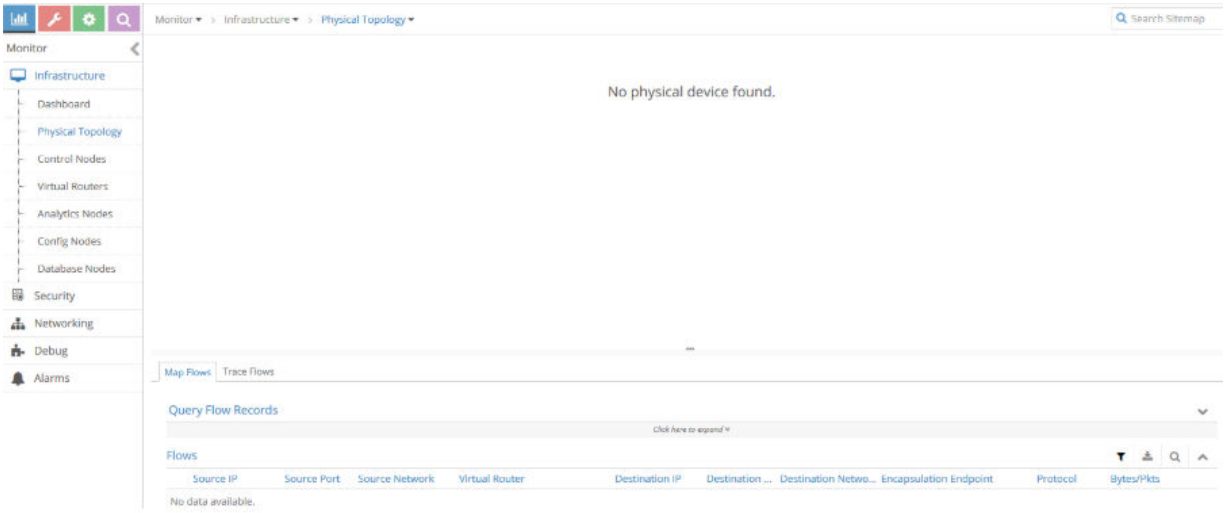
Figure 154: Analytics Alarm Disabled - Configure > Alarms



Analytics SNMP Feature Enabled

Figure 155 on page 329 displays the Physical Topology option in the Monitor menu.

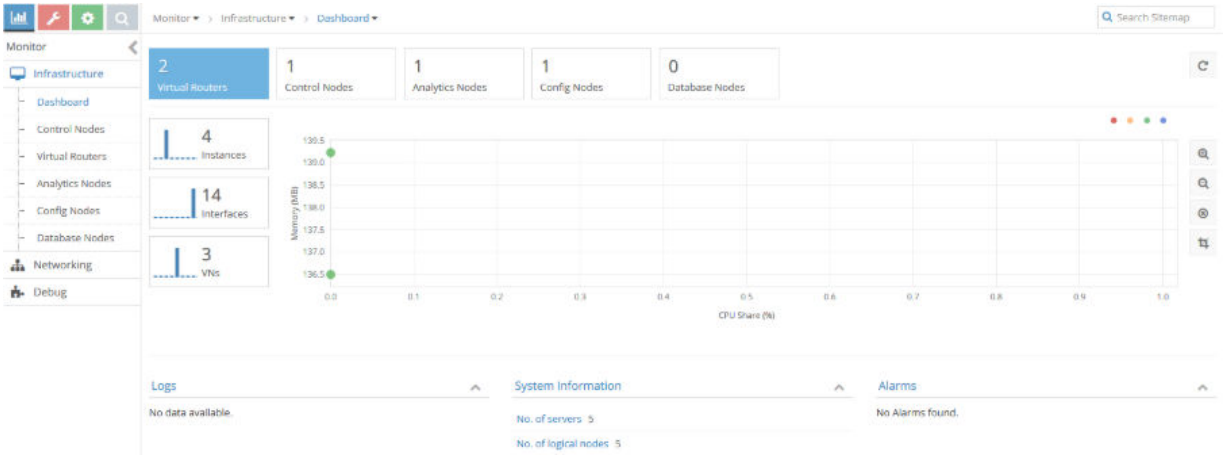
Figure 155: Analytics SNMP Feature Enabled - Physical Topology Menu Available



Analytics SNMP Feature Disabled

If the alarm analytics component is not provisioned, then Contrail Web UI does not display the Physical Topology menu option.

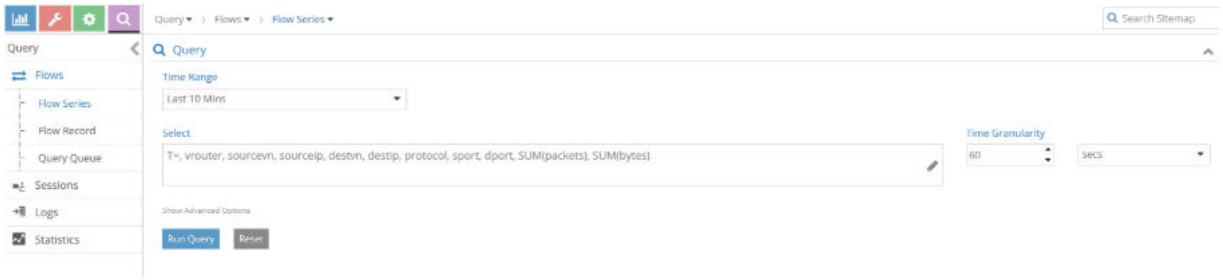
Figure 156: Analytics SNMP Feature Disabled - Physical Topology Menu Not Available



Analytics Database Enabled

If analytics database is provisioned, then Contrail Web UI displays the Query page.

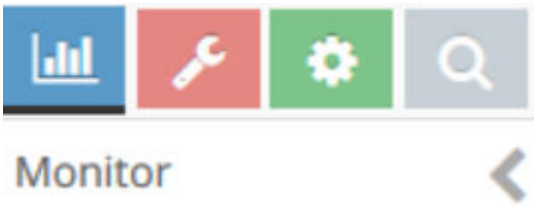
Figure 157: Analytics Database Enabled - Query Page Available



Analytics Database Disabled

If analytics database is not provisioned, then Contrail Web UI should not display the Query page. Query page logo is unavailable to launch Query page.

Figure 158: Analytics Database Disabled - Query Page Logo Not Available



Tripleo Provisioning

Multi-Nodes Contrail Controller Topology

In order to enable or disable the Contrail analytics optional components, TripleO templates have to be modified.

- In ContrailAnalytics role, **ContrailAnalyticsSnmp**, and **ContrailAnalyticsAlarm** resources can be removed:

```
- OS::TripleO::Services::ContrailAnalytics
- OS::TripleO::Services::ContrailAnalyticsSnmp
- OS::TripleO::Services::ContrailAnalyticsAlarm
```

- ContrailAnalyticsDatabase role can also be removed (not selected using ContrailAnalyticsDatabaseCount = 0) into a rollout as this role is deploying only **ContrailAnalyticsDatabase** resource:

```
- OS::TripleO::Services::ContrailAnalyticsDatabase
```

- ContrailController role is kept unchanged.

Single Node Contrail Controller Topology

In order to enable or disable the Contrail analytics optional components, TripleO templates have to be modified. In ContrailController role, **ContrailAnalyticsSnmp**, **ContrailAnalyticsAlarm**, and **ContrailAnalyticsDatabase** resources can be removed, other contrail resources are kept:

```
- name: ContrailController
    - OS::TripleO::Services::ContrailAnalytics
    - OS::TripleO::Services::ContrailAnalyticsAlarm
```

```
    - OS::TripleO::Services::ContrailAnalyticsDatabase
    - OS::TripleO::Services::ContrailAnalyticsSnmp
    - OS::TripleO::Services::ContrailCertmongerUser
    - OS::TripleO::Services::ContrailConfig
    - OS::TripleO::Services::ContrailConfigDatabase
    - OS::TripleO::Services::ContrailControl
    - OS::TripleO::Services::ContrailWebui
```

TripleO Template Update

TripleO templates were updated in June 2020 to allow disabling the provisioning of Contrail analytics components.

Earlier Contrail TripleO templates have to be patched in order to replace `docker/services/contrail/contrail-base.yaml` file in which optional analytics component provision is hardcoded:

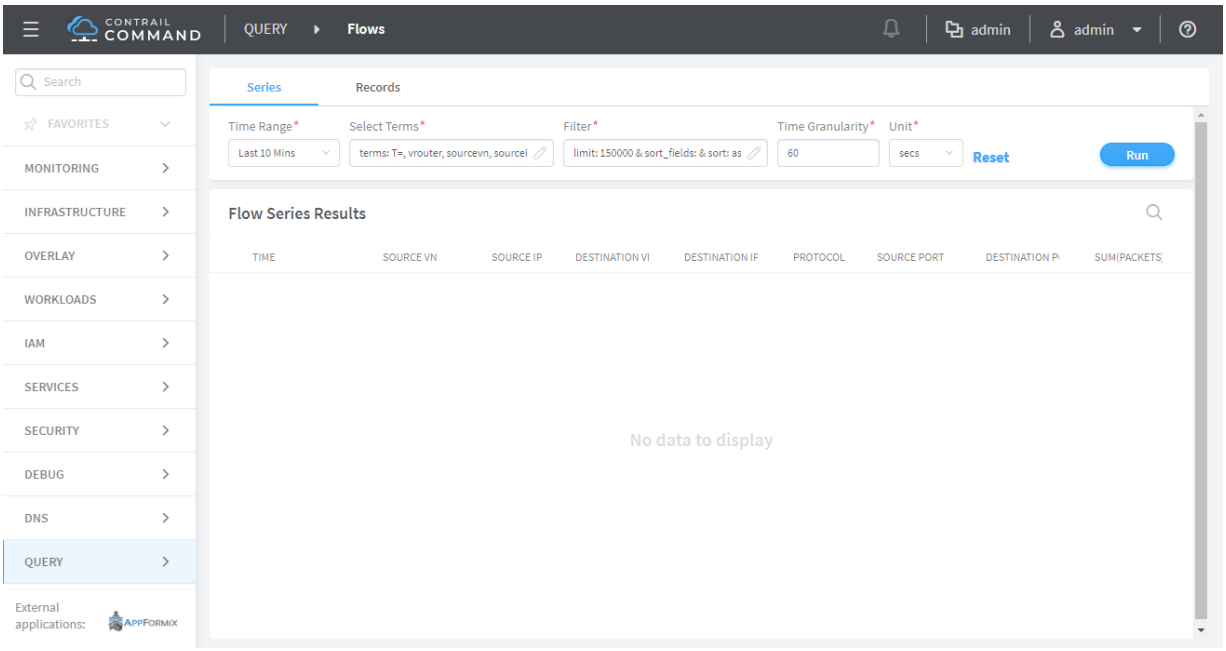
```
...
                ANALYTICS_ALARM_ENABLE: 'False'
                ANALYTICS_SNMP_ENABLE: 'True'
                ANALYTICSDB_ENABLE: 'True'
...
```

Appendix

Contrail Command UI

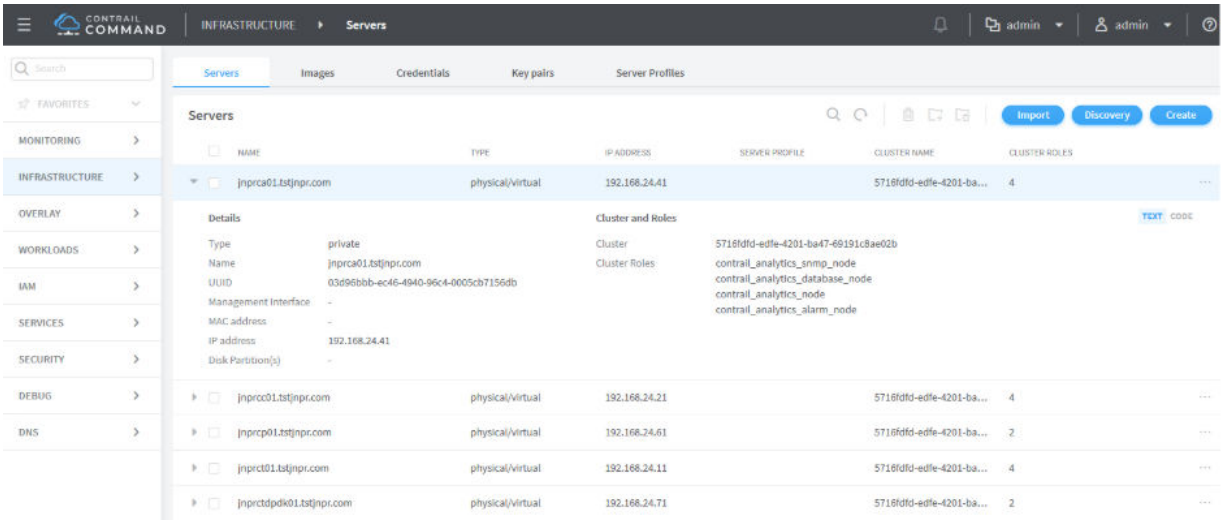
The disabled roles and charts are visible on the Query page but they are not operational.

Figure 159: Query Page Visible in Dashboard



Regardless that the alarm, SNMP, and database analytics roles have been disabled, they are still reported by Contrail Command.

Figure 160: Disabled Roles Still Visible in Contrail Command



The following five charts will always display empty.

Figure 161: Empty Charts in Compute Nodes

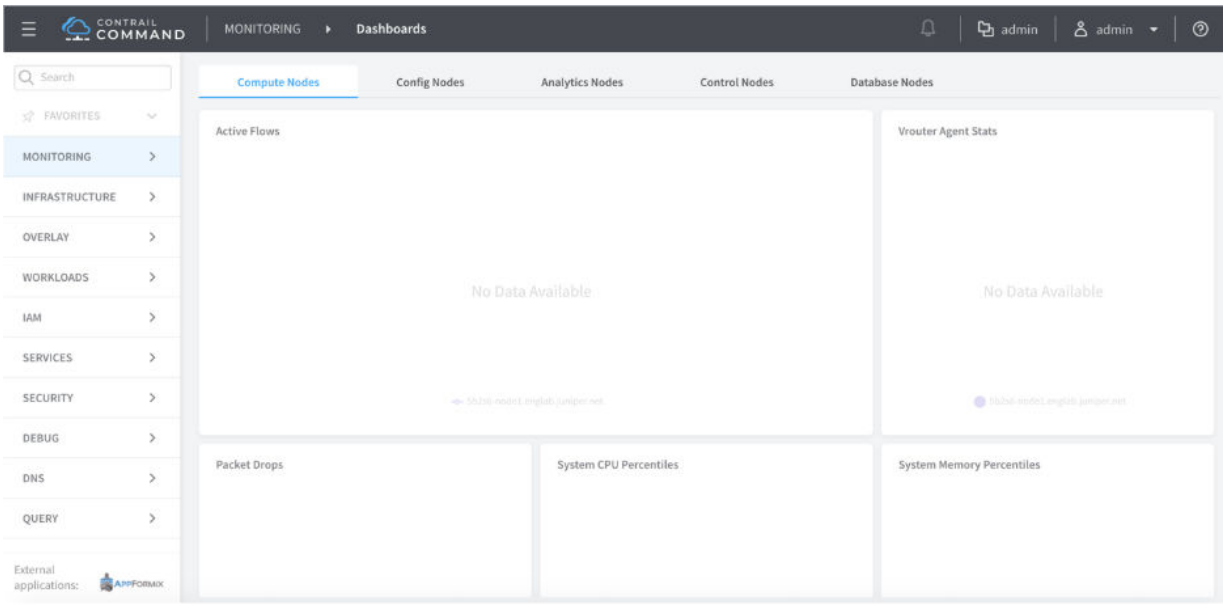


Figure 162: Empty Charts in Config Nodes

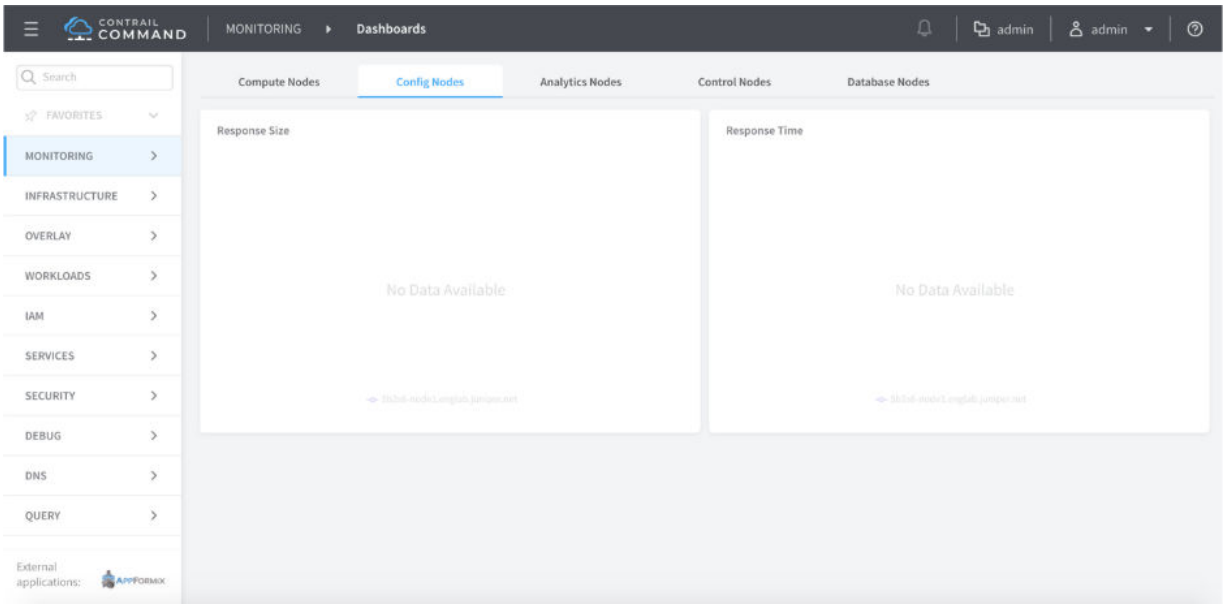


Figure 163: Empty Charts in Analytics Nodes

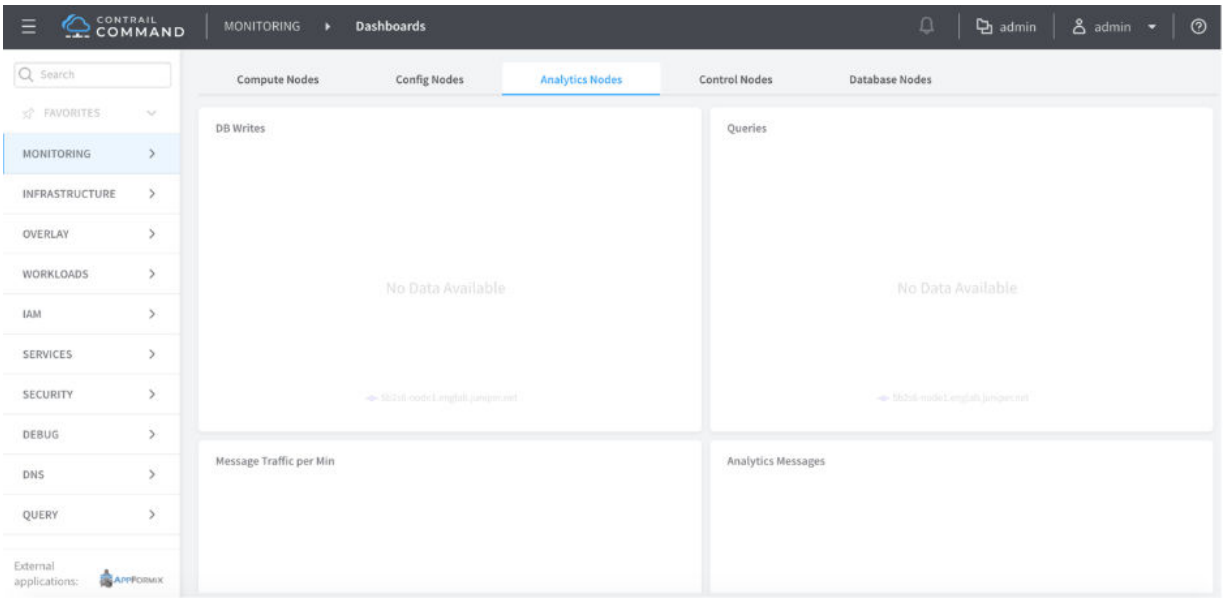


Figure 164: Empty Charts in Control Nodes

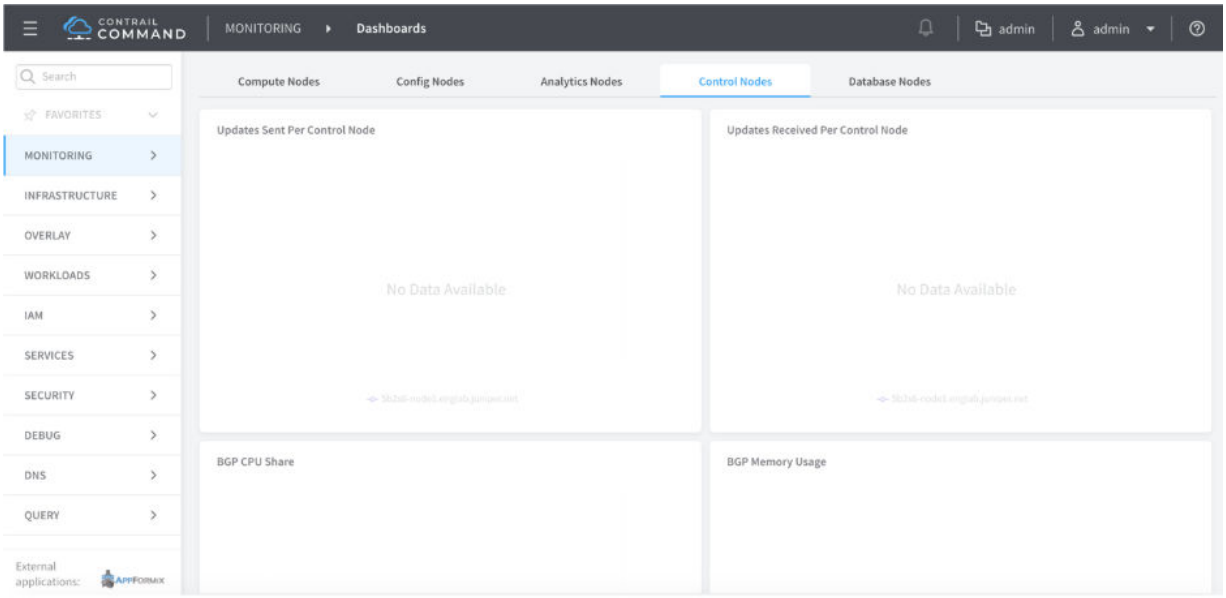
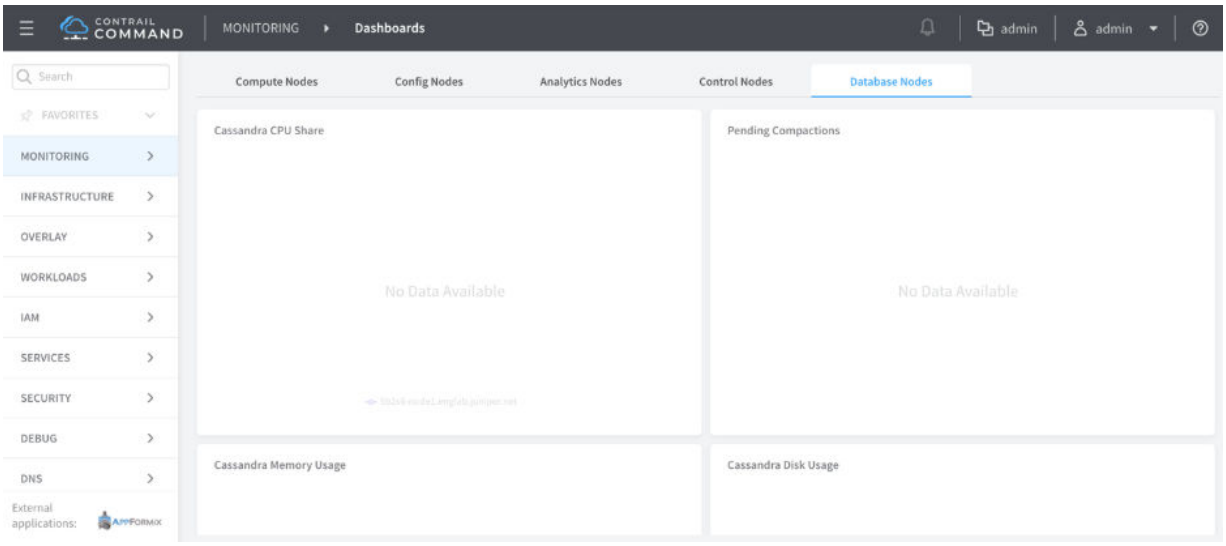
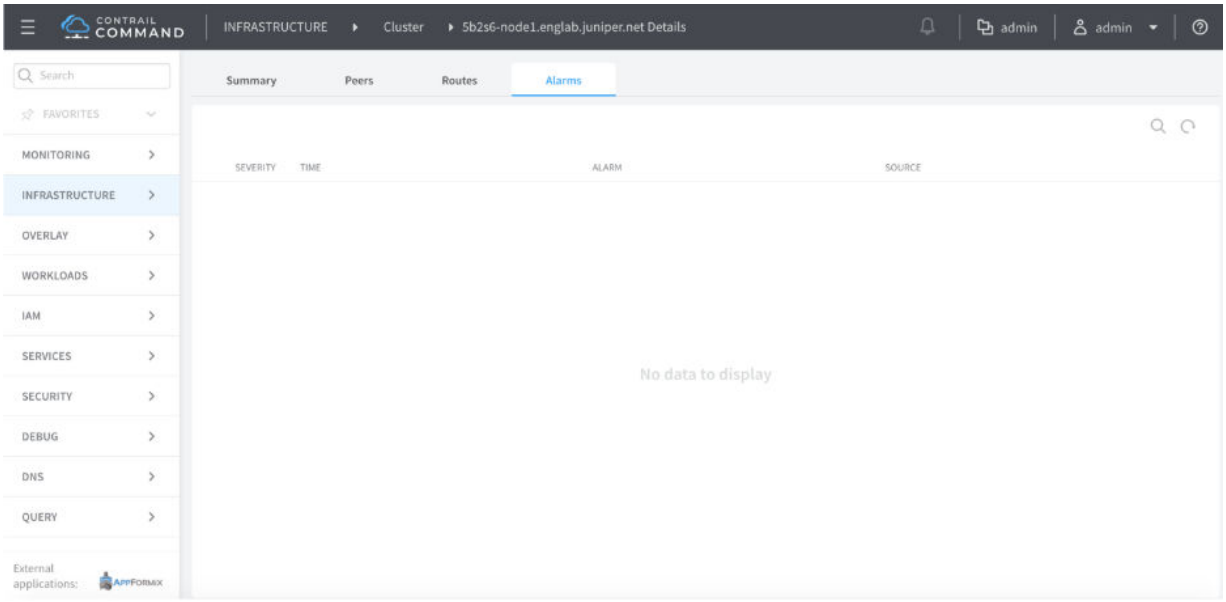


Figure 165: Empty Charts in Database Nodes



The Alarms page displays alarms pulled from the Contrail `analytics_alarm` component. When the `analytics_alarm` component is disabled, the Alarms page will always display no data.

Figure 166: Empty Alarms Page



Change History Table

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

Release	Description
2008	TripleO templates were updated in June 2020 to allow disabling the provisioning of Contrail analytics components.

CHAPTER 6

Contrail Insights in Contrail Command

IN THIS CHAPTER

- Contrail Insights Overview | 338
- Contrail Insights Flows in Contrail Command | 339
- Viewing Telemetry KPI Alarms for Fabric Devices and Ports | 353
- Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command | 364
- Adding or Deleting sFlow Collector Nodes by Modifying instances.yml | 378
- Configuring Contrail Insights Alarms using Contrail Command | 381
- Configuring Instances in Contrail Insights | 406
- Viewing Cluster Node Details and Metric Values | 412

Contrail Insights Overview

Contrail Insights 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.

Contrail Insights 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 Contrail Insights features are supported in Contrail Command:

- Installing Contrail Insights using Contrail Command
- Configuring Contrail Insights Alarms using Contrail Command
- Configuring Instances in Contrail Insights

- Viewing Cluster Node Details and Metric Values

NOTE: For information about installing Contrail Insights, see the *Contrail Installation and Configuration Guide*. For more information about Contrail Insights, see the Contrail Insights User Guide.

RELATED DOCUMENTATION

How to Install Contrail Command and Provision Your Contrail Cluster

Configuring Contrail Insights Alarms using Contrail Command | **381**

Configuring Instances in Contrail Insights | **406**

Viewing Cluster Node Details and Metric Values | **412**

Contrail Insights Flows in Contrail Command

IN THIS SECTION

- Configuring Contrail Insights Flows from Contrail Command | **339**
- Configuring Contrail Insights Flows During Fabric Onboarding | **340**
- Configuring Contrail Insights Flows by Assigning Telemetry and sFlow Profiles to Devices | **341**
- Removing a Telemetry Profile | **351**

These topics describe how to configure Contrail Insights Flows from Contrail Command.

Configuring Contrail Insights Flows from Contrail Command

Starting with Contrail Networking Release 1910, Contrail Insights Flows is integrated in the Contrail Command UI. Contrail Insights Flows enables you to view telemetry information for the devices in a Contrail-managed data center fabric. With the addition of this feature, Contrail Command acts as a single pane of glass where you can access the features of both Contrail Networking and telemetry feature of Contrail Insights, providing you a unified telemetry experience.

For Contrail Networking Release 1910, the flow collector provisioning is disabled by default in the provisioning wizard. To enable flow collector provisioning, log in to the `contrail_command` container and edit the **/usr/share/contrail/public/feature-list.json** file. Set the value for `cluster_user.xflow` to `true`.

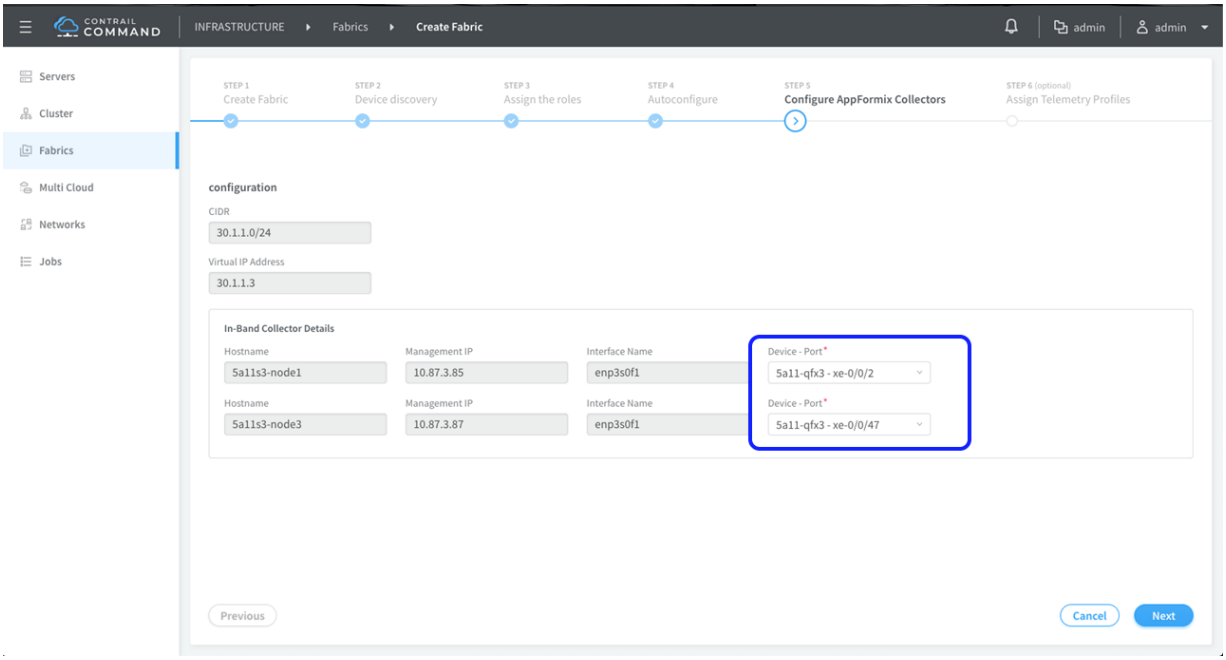
In Contrail Networking Release 1911 and later, the flow collector provisioning is enabled by default.

Contrail Networking Release 1911 supports provisioning of both in-band and out-of-band collectors. You can configure Contrail Insights Flows during initial setup, during fabric onboarding, or by creating and assigning telemetry and sFlow profiles to devices. For more information about configuring Contrail Insights Flows during Contrail Command installation, see Installing Contrail Insights and Contrail Insights Flows using Contrail Command in the *Contrail Installation and Upgrade Guide*.

Configuring Contrail Insights Flows During Fabric Onboarding

You use this procedure to provision an In-Band collector during the fabric onboarding workflow. Here, the information that you entered during initial setup is displayed and you can specify the **Device Port** that you want to associate with the flow collector.

Figure 167: Configure Device Port



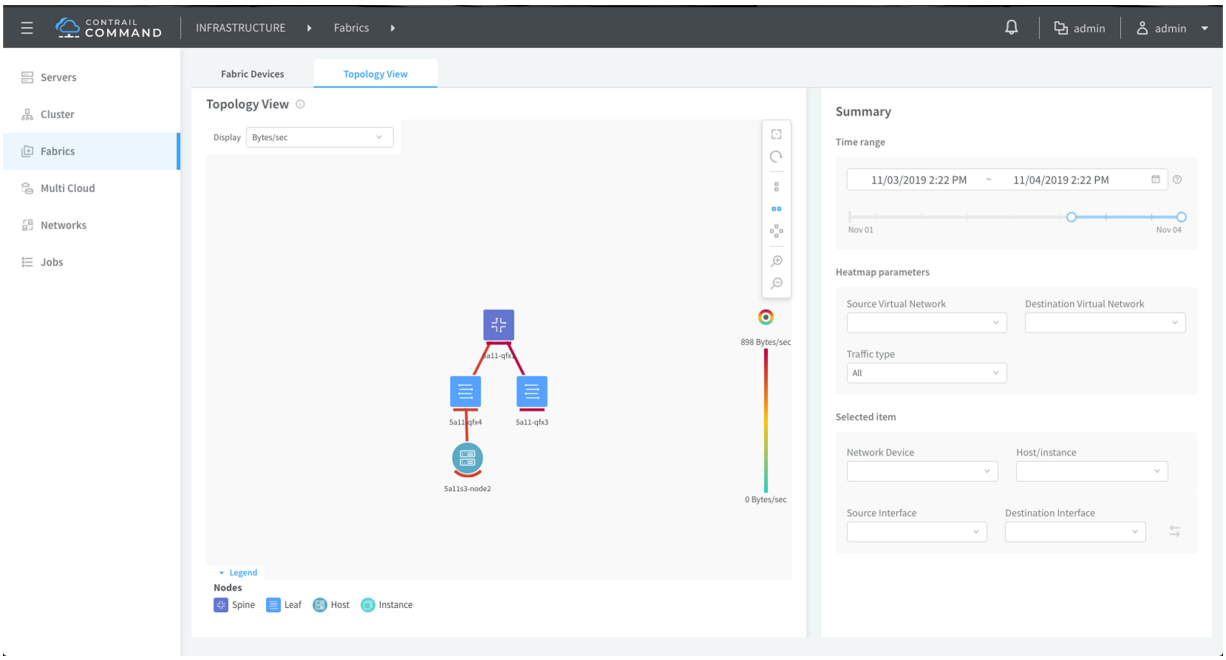
When you click **Next**, configuration similar to the following is pushed to the device:

```
set groups __contrail_underlay_infra_bms_access interfaces xe-0/0/2 unit 0 family ethernet-
switching vlan members elemetry_infra_network_ipam-vlan
set groups __contrail_underlay_infra_bms_access interfaces xe-0/0/47 unit 0 family ethernet-
```

```
switching vlan members elemetry_infra_network_ipam-vlan
set groups __contrail_underlay_infra_bms_access interfaces irb unit 11 family inet address
30.1.1.1/24
set groups __contrail_underlay_infra_bms_access vlans elemetry_infra_network_ipam-vlan vlan-id 11
set groups __contrail_underlay_infra_bms_access vlans elemetry_infra_network_ipam-vlan l3-
interface irb.11
```

After fabric provisioning is complete, you can view the flow data from the **Infrastructure** > **Fabrics** > *Fabric Name* > **Topology View** page.

Figure 168: Topology View



Configuring Contrail Insights Flows by Assigning Telemetry and sFlow Profiles to Devices

This topic describes how to provision Contrail Insights Flows and assign telemetry profiles after setting up Contrail Command and discovering devices.

The benefit of assigning telemetry profiles is that you can monitor the health of different devices and their interfaces from Contrail Command after the telemetry profile gets configured on these devices.

NOTE: If telemetry profiles are not configured, there will be “No data” for the “top talkers” in the Contrail Command Top-N-View. See "Top N View in Contrail Command" on page 280 .

After Contrail Command is set up and devices are discovered, you can attach telemetry profiles to devices. You can attach only one telemetry profile per device. Each telemetry profile is linked to sub-profile(s). The telemetry profile can contain all types of sub-profiles but only one instance each of the sFlow, gRPC, Netconf, or SNMP sub-profiles. You can either link a telemetry profile to an existing sub-profile or create a new sub-profile while creating the telemetry profile.

Default sFlow profiles and telemetry profiles are predefined in the system when you bring up the cluster. You cannot edit or delete these default profiles. However, you can create custom profiles and associate them to the telemetry profile.

The sFlow monitoring technology collects samples of network packets and sends them to a monitoring station called a *collector*. The sFlow technology implements two sampling mechanisms:

- Packet-based sampling—Samples one packet out of a specified number of packets from an interface enabled for sFlow technology.
- Time-based sampling—Samples interface statistics (counters) at a specified interval from an interface enabled for sFlow technology.

Contrail Networking Release 2011 supports gRPC, Netconf, and SNMP protocol-based telemetry profiles. Contrail Insights collects key performance indicators (KPIs) from network devices using preconfigured values to monitor the fabric health.

To view the health of your fabric devices, ports, and any alerts associated with exceeding KPI thresholds, navigate to **Infrastructure** > **Fabrics** > *<Fabric Name>*.

The default sFlow telemetry profiles are:

- sflow-access-interfaces—Indicates that sFlow is enabled on all the access interfaces on the device.
- sflow-fabric-interfaces—Indicates that sFlow is enabled on all the fabric interfaces.
- sflow-all-interfaces—Indicates that sFlow is enabled on all the interfaces on the device that has an sFlow profile attached to it.

The default protocol-based telemetry profiles are:

- grpc-default-profile—Indicates that the health parameters for health/environment, interface, and control plane sensors are enabled for monitoring. This profile includes an Allowed Clients List with a default value of `0.0.0.0/0`. See Figure 173 on page 347 and Table 60 on page 348 .

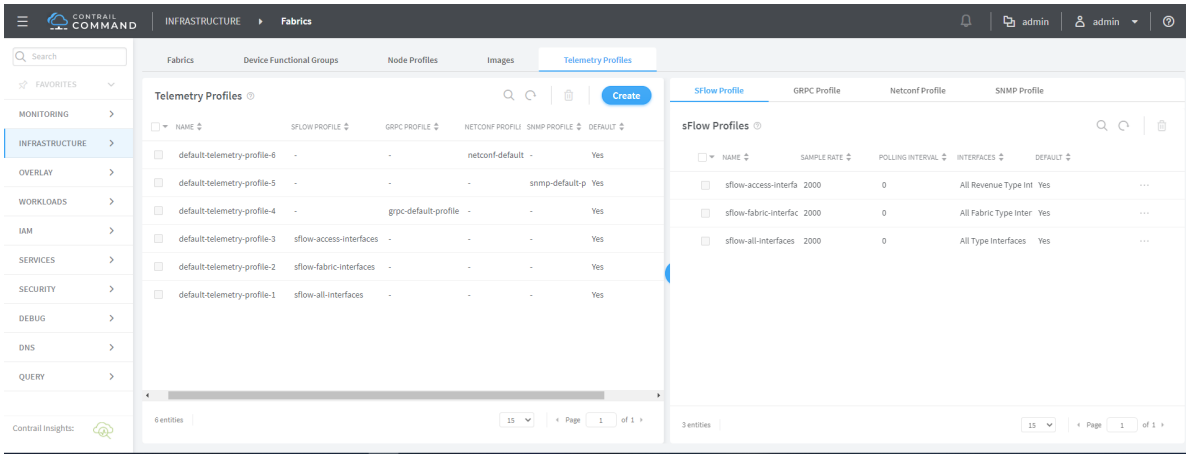
- netconf-default-profile—Indicates that the health parameters for health/environment, interface, and control plane sensors are enabled for monitoring.
- snmp-default-profile—Indicates that the health parameters for health/environment, interface, and control plane sensors are enabled for monitoring.

You can apply default profiles to network devices and generate alerts based on predefined KPIs and preconfigured alert generation rules.

To create a telemetry profile:

1. Log in to Contrail Command UI and navigate to **Infrastructure** > **Fabrics**.
2. Click the **Telemetry Profiles** tab.

Figure 169: Telemetry Profiles Tab



Click profile tabs (**sFlow**, **GRPC**, **Netconf**, **SNMP**) to view existing profiles.

3. Click **Create** to define a new telemetry profile.

   You can assign one or more profiles to the telemetry profile by selecting existing profiles from the list(s). You can create new profile(s) by clicking **Create New** and then assign it to the telemetry profile.

   Figure 170 on page 344 shows assigning an existing sFlow profile. Mouse over the sFlow name to view the profile details.

Figure 170: Assign an Existing sFlow Profile

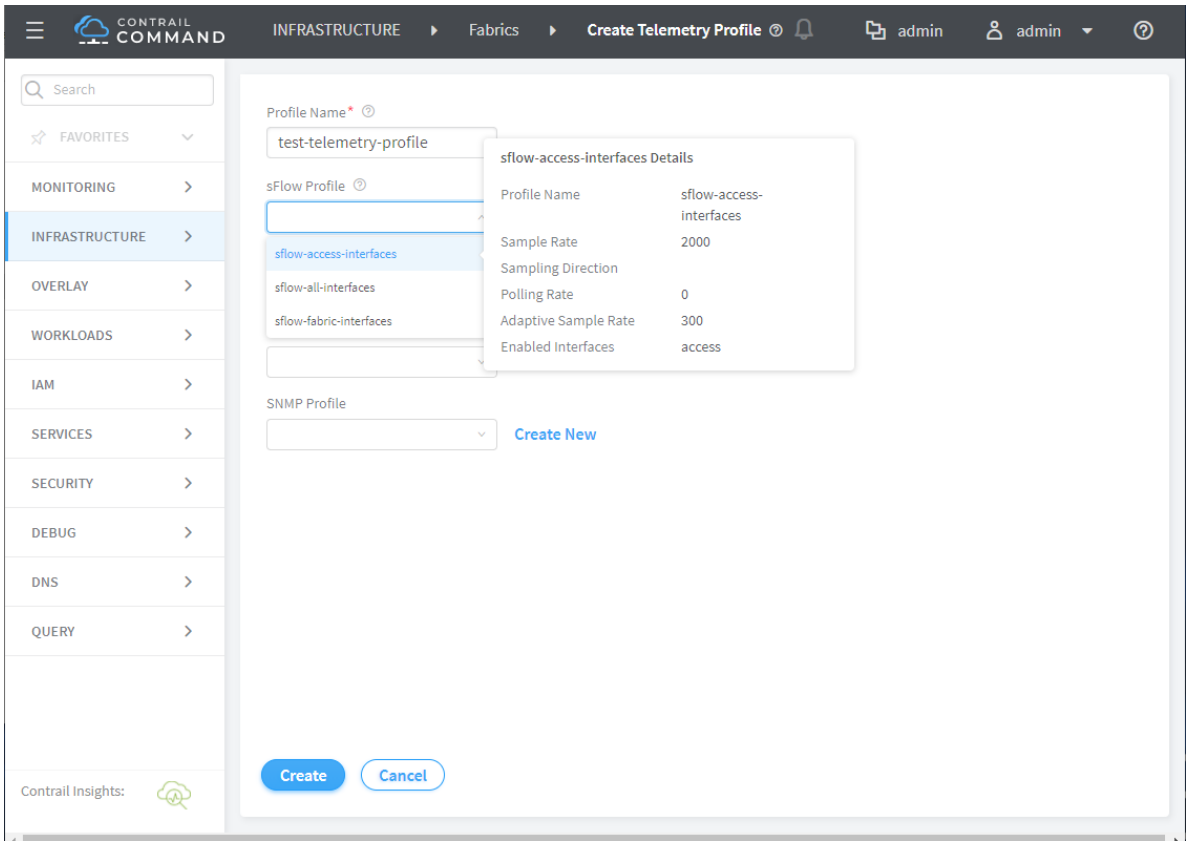
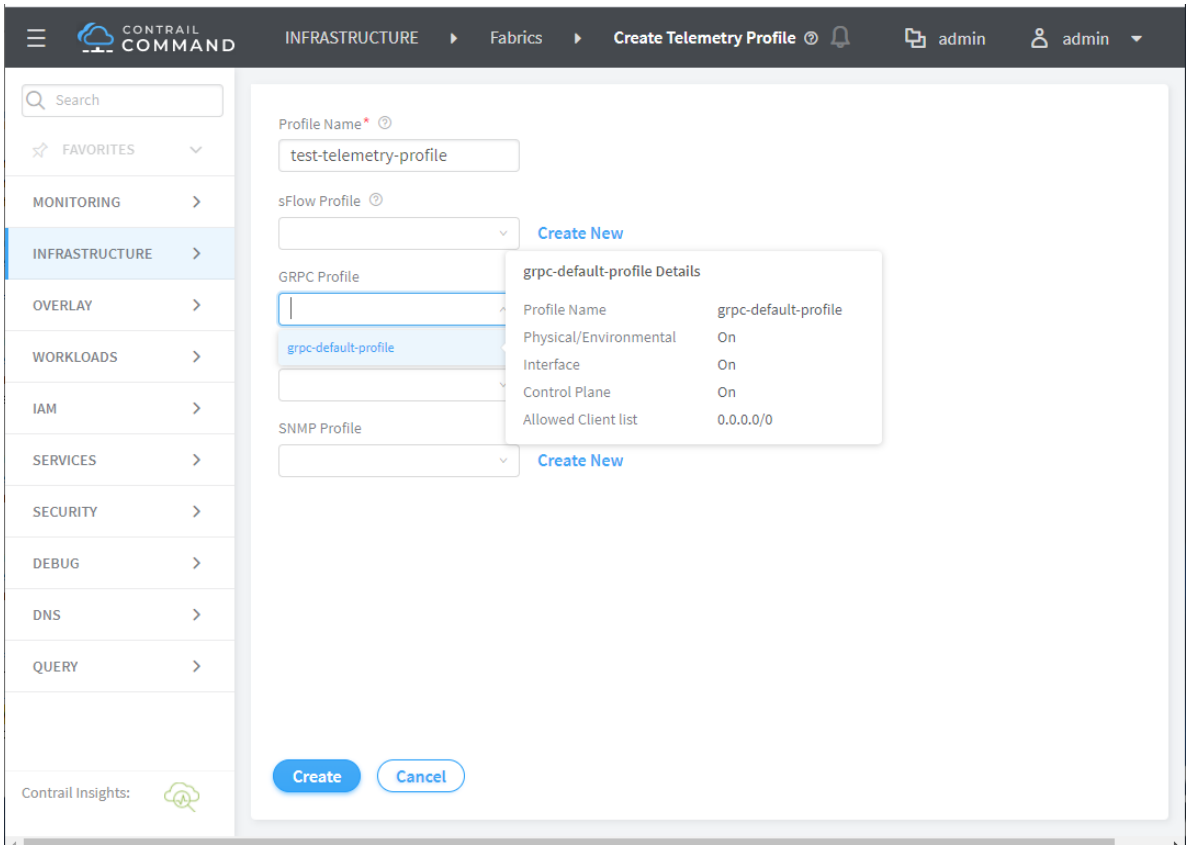


Figure 171: Assign an Existing gRPC Profile



4. (Optional) Click **Create New** next to the profile fields to create a new profile.

   Figure 172 on page 346 shows an example of creating a sFlow profile.

Figure 172: Create New sFlow Profile

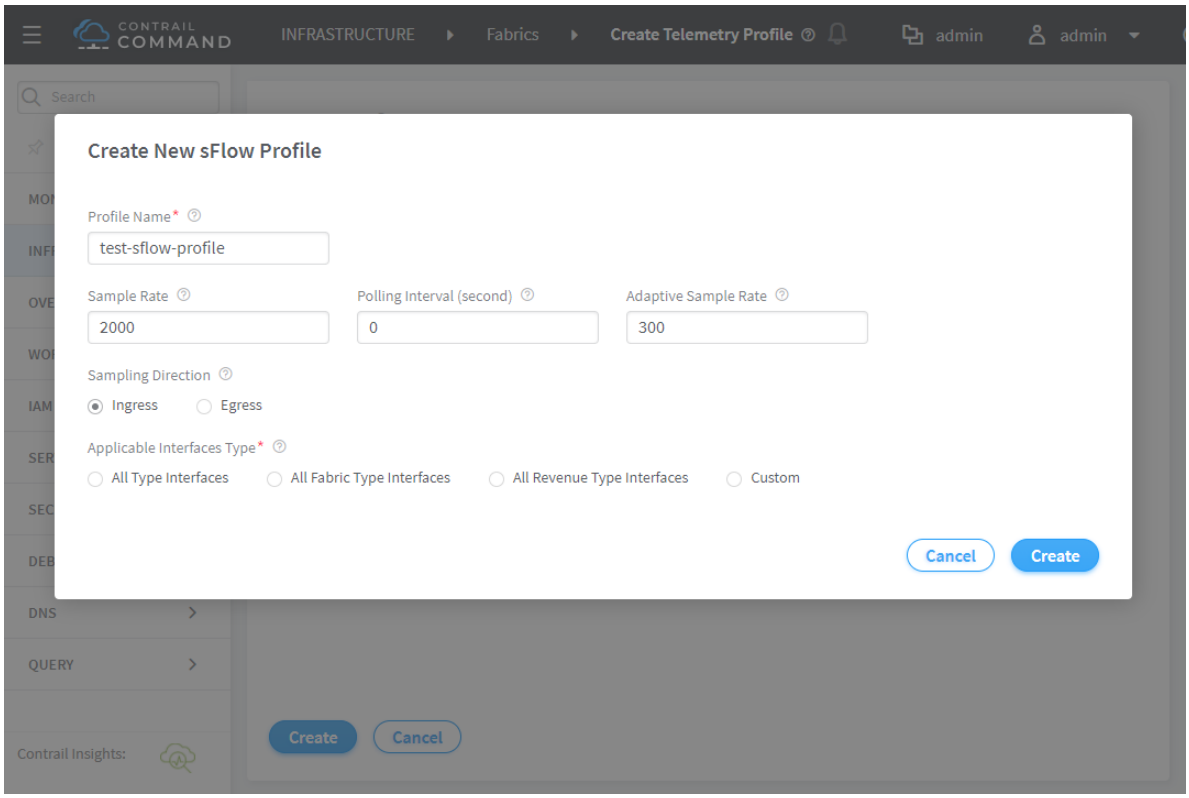


Table 59: sFlow Profile Fields

Field	Description
Profile Name	Enter a name for the profile you are creating.
Sample Rate	The configured number of egress or ingress packets out of which one packet is sampled. For example, with the default sample rate of 2000, meaning one packet out of 2000, is sampled.
Polling Interval (second)	Configure the interval (in seconds) that the device waits between port statistics update messages. Default should be set to 0 (recommended).
Adaptive Sample Rate	Process of monitoring the overall incoming traffic rate on the network device and providing intelligent feedback to interfaces to dynamically adapt the sampling rates on interfaces on the basis of traffic conditions. The default is one out of every 300 packets.

Table 59: sFlow Profile Fields *(Continued)*

Field	Description
Sampling Direction	Packets are sampled either at the ingress or egress interfaces for a given network path flow.
Applicable Interfaces Type	Select the type of interface you want to monitor.

Configure a gRPC, Netconf, or SNMP profile to monitor the health of your network and generate alarms to identify any anomalies. Alarms are generated based on the collected metrics and preconfigured thresholds in the alarm rules.

Figure 173 on page 347 shows an example of creating a gRPC profile.

Figure 173: Create New gRPC Profile

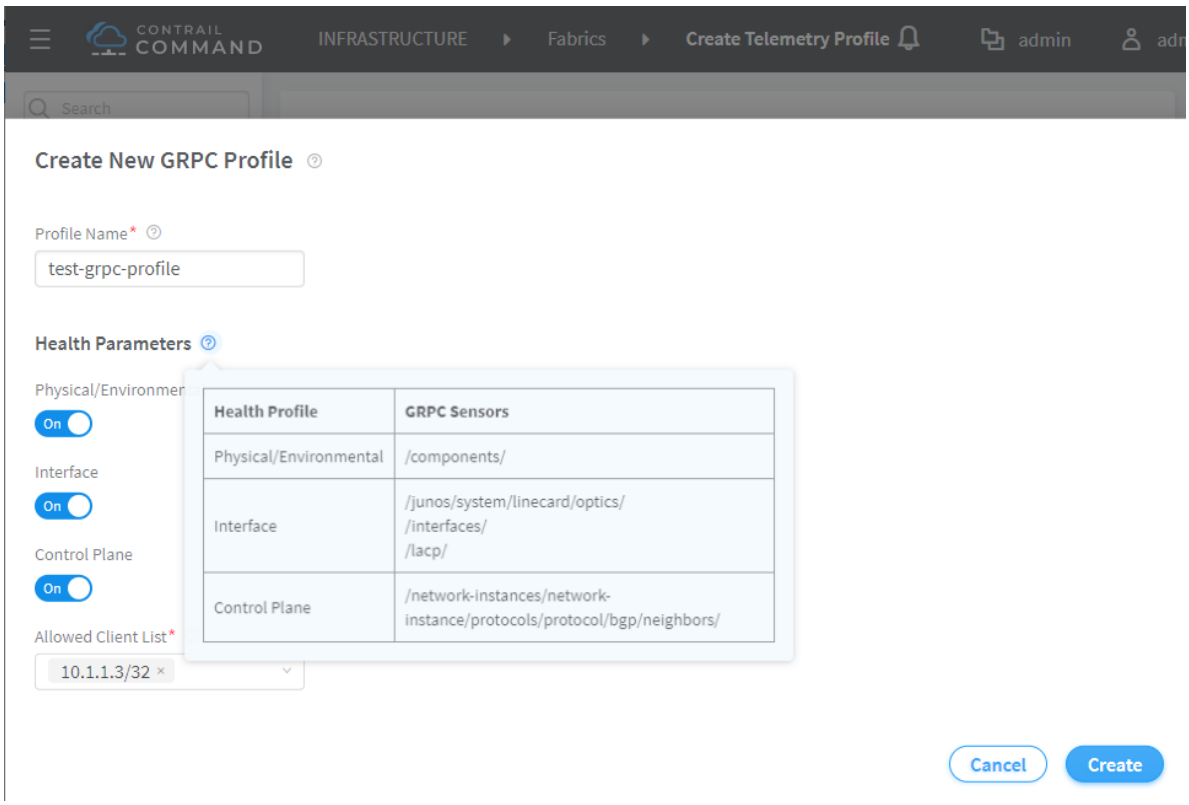


Table 60: gRPC, Netconf, and SNMP Profile Fields

Field	Description
Profile Name	Enter a name for the gRPC, Netconf, or SNMP profile you are creating.
Health Parameters • Physical/Environmental • Interface • Control Plane	Mouse over the **?** to view a table listing the applicable sensors, commands, or MIBs for physical/environmental, interface, and control plane monitoring. By default, all three settings are **On**, which is recommended. See Figure 173 on page 347 .
Allowed Client List	**NOTE**: This field applies to grpc-default-profile only. Default value is `0.0.0.0/0`. Prepopulated subnets are from the Contrail Insights node. Add additional IP addresses (using the CIDR format `1.1.1.1/32`) and/or subnets and Enter. You can also copy and paste entries.

5. Attach new sFlow profile to the telemetry profile.

Figure 174: Assign New sFlow Profile to Telemetry Profile

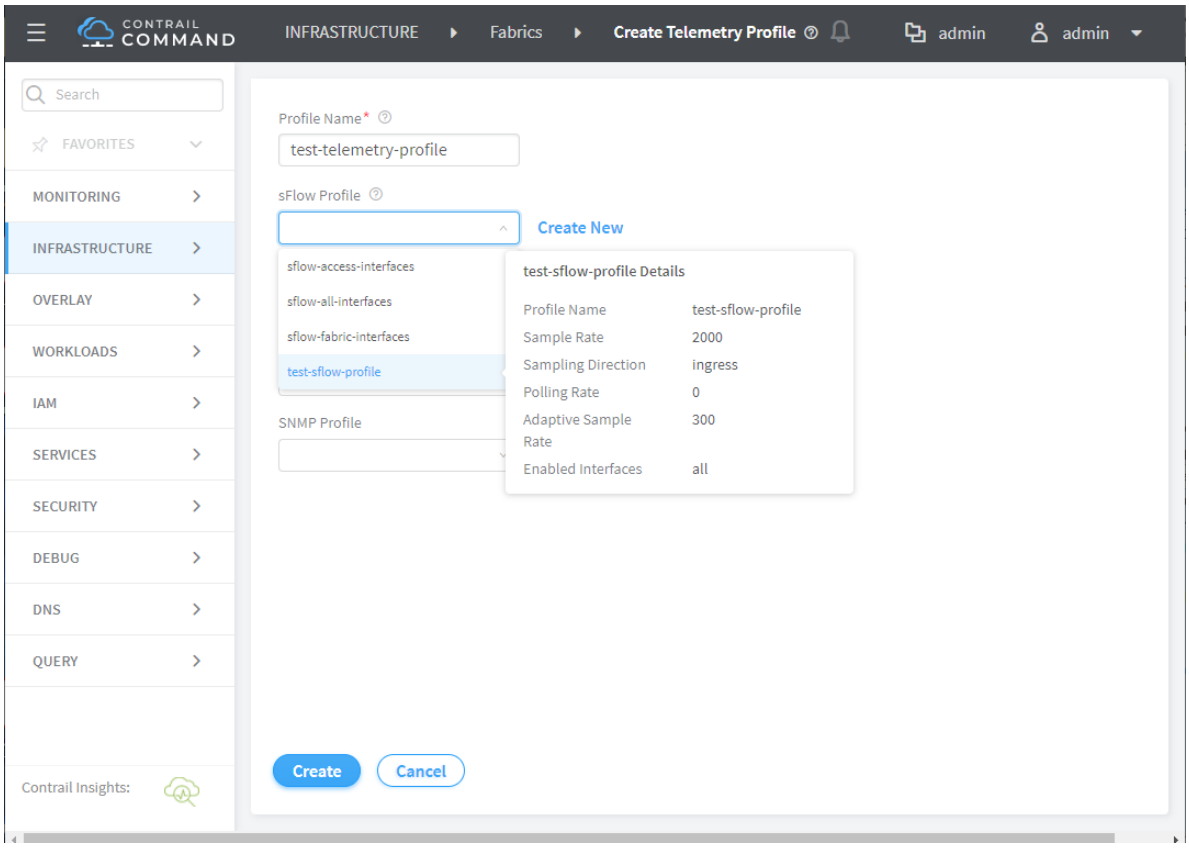
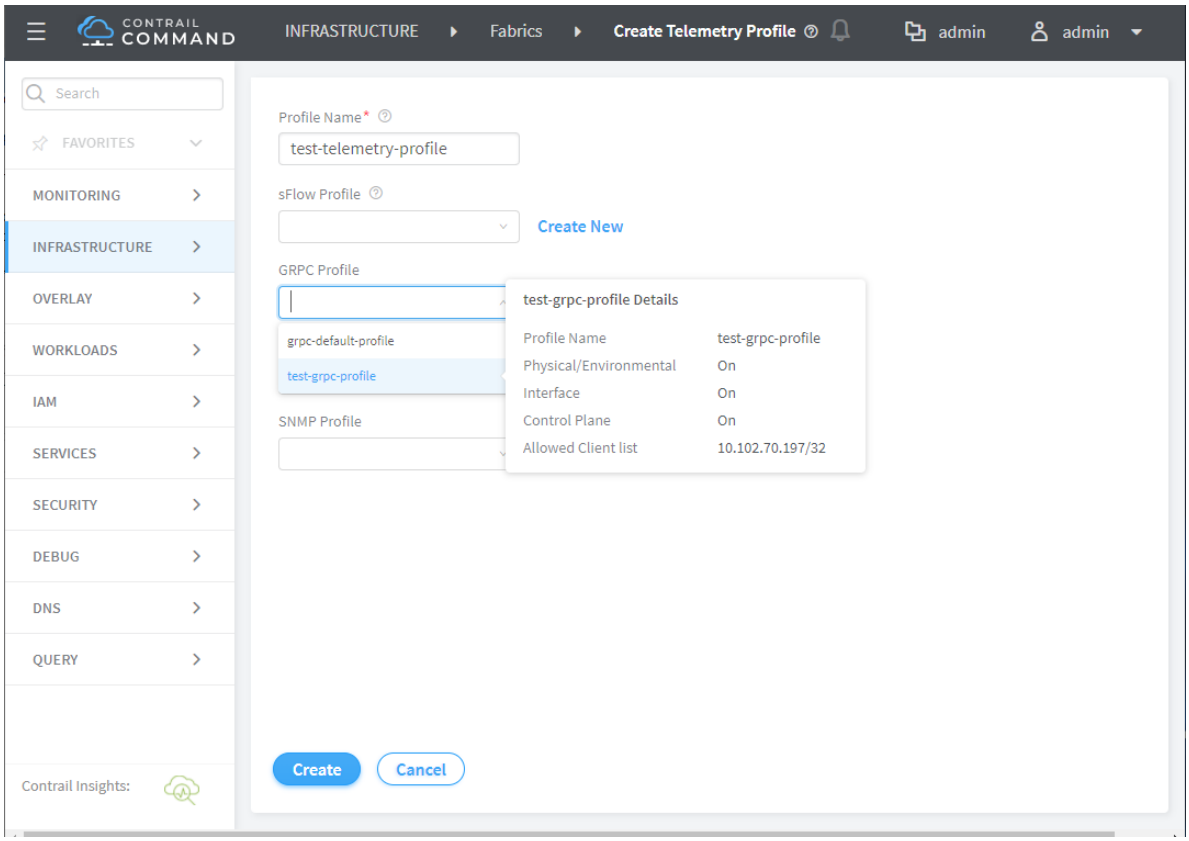


Figure 175: Assign New gRPC Profile to Telemetry Profile



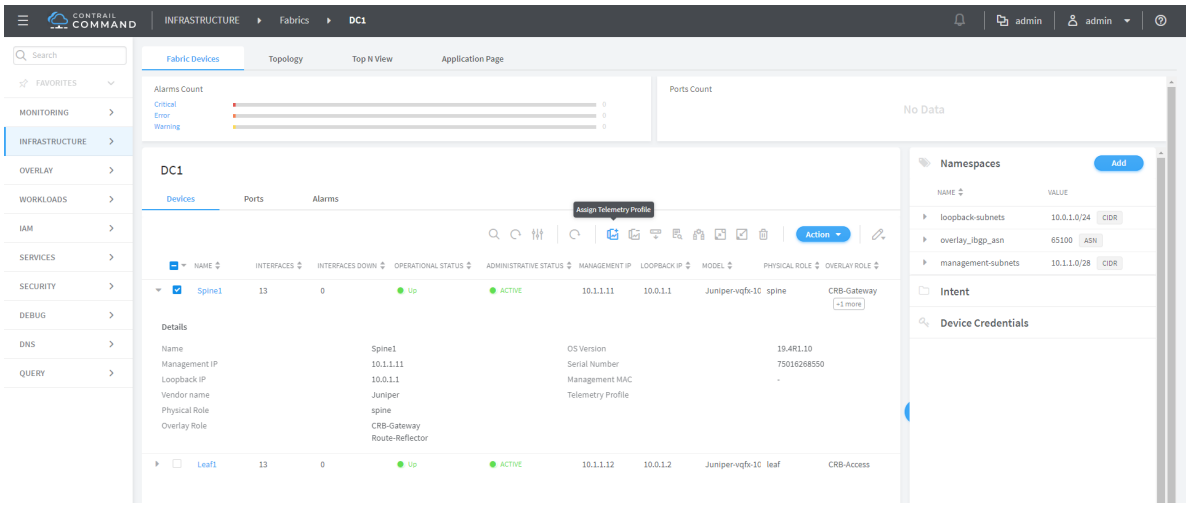
6. Click **Create**.

   Continue with the remaining steps to assign your telemetry profile to a device.

7. Click **Infrastructure** > **Fabrics** > ***Fabric Name*** to assign a telemetry profile to one or more devices in a fabric.

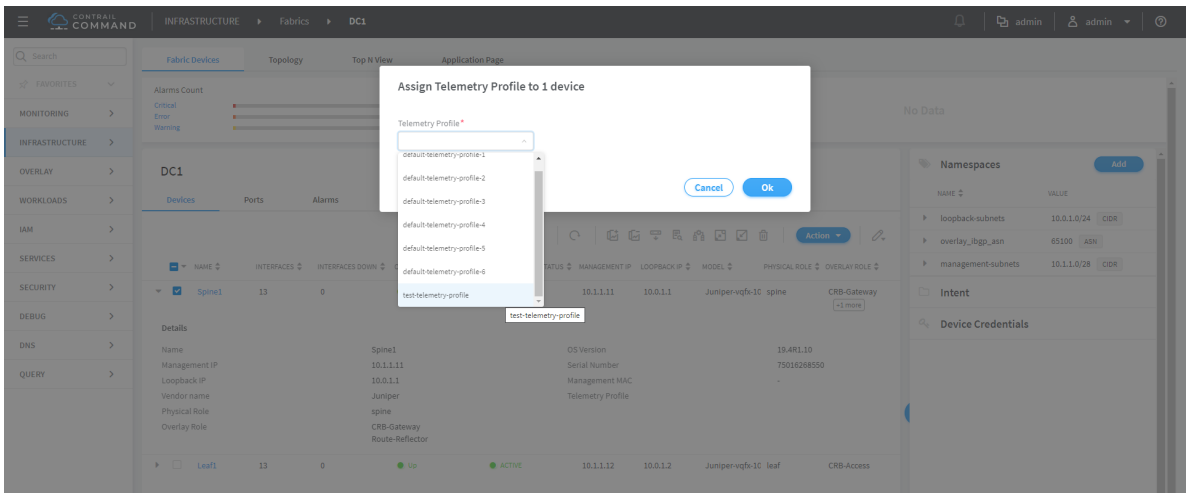
8. Select the device or devices for which you want to assign telemetry profile and click **Assign Telemetry Profile** as shown in Figure 176 on page 351 .

Figure 176: Assign Telemetry Profile



9. Select a telemetry profile from the list and click **Ok** as shown in Figure 177 on page 351 .

Figure 177: Select Telemetry Profile to Assign to Device



The selected telemetry profile is now assigned to the device.

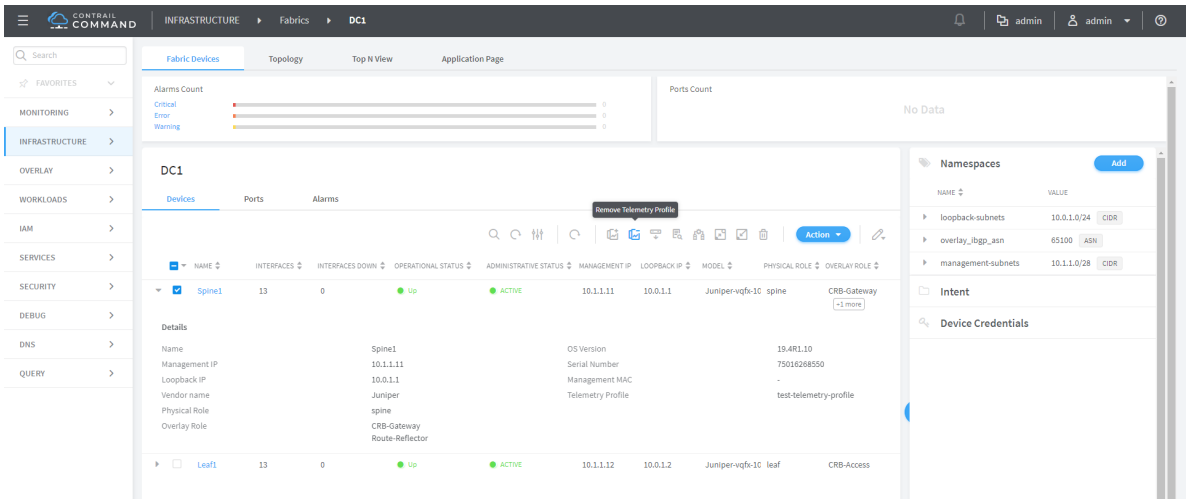
To view the health of your fabric, navigate to **Infrastructure** > **Fabrics** > *<Fabric Name>*.

Removing a Telemetry Profile

To remove a telemetry profile assigned to a device:

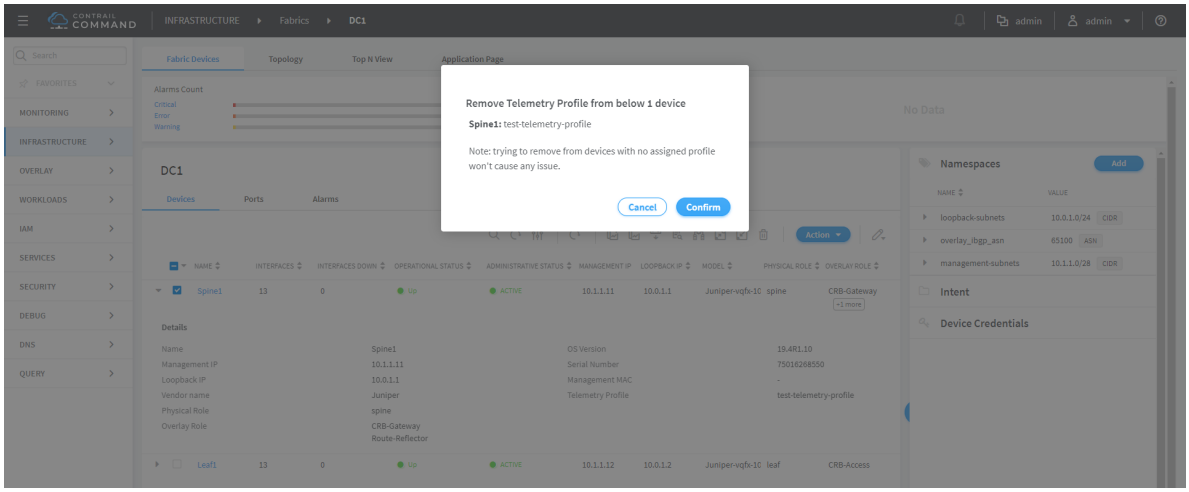
1. Navigate to **Infrastructure** > **Fabrics** > *<Fabric Name>*.
2. Click **Remove Telemetry Profile** as shown in Figure 178 on page 352 .

Figure 178: Remove Telemetry Profile



3. Click **Confirm** to remove the telemetry profile attached to the device.

Figure 179: Confirm Removing Telemetry Profile



Change History Table

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

Release	Description
2011	Contrail Networking Release 2011 supports gRPC, Netconf, and SNMP protocol-based telemetry profiles. Contrail Insights collects key performance indicators (KPIs) from network devices using preconfigured values to monitor the fabric health.

1911	Contrail Networking Release 1911 supports provisioning of both in-band and out-of-band collectors.
1910	Starting with Contrail Networking Release 1910, Contrail Insights Flows is integrated in the Contrail Command UI.

RELATED DOCUMENTATION

- Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command | **364**
- Viewing Telemetry KPI Alarms for Fabric Devices and Ports | **353**
- Top N View in Contrail Command | **280**
- Adding or Deleting sFlow Collector Nodes by Modifying instances.yml | **378**

Viewing Telemetry KPI Alarms for Fabric Devices and Ports

IN THIS SECTION

- Fabric Devices | **353**
- Ports | **357**
- Alarms | **358**
- Alarms Count, Ports Count, and Device Overview | **359**
- Configure and Assign Telemetry Profiles from Fabric Overview | **361**

In Contrail Networking Release 2011, the Fabrics page displays additional detail about the health of your devices and interfaces, gathering data from telemetry metrics and profiles configured on these devices. These pages will show if any key performance indicators (KPIs) have crossed a threshold value. You can also configure metrics and assign telemetry profiles from the Fabrics Overview page. See Figure 189 on page 362 .

Fabric Devices

To view fabric devices:

1. Navigate to **Infrastructure** > **Fabrics**.

   The Fabrics page displays. An error icon displays next to the fabrics name if there are critical or major alarms present.

Figure 180: Infrastructure > Fabrics

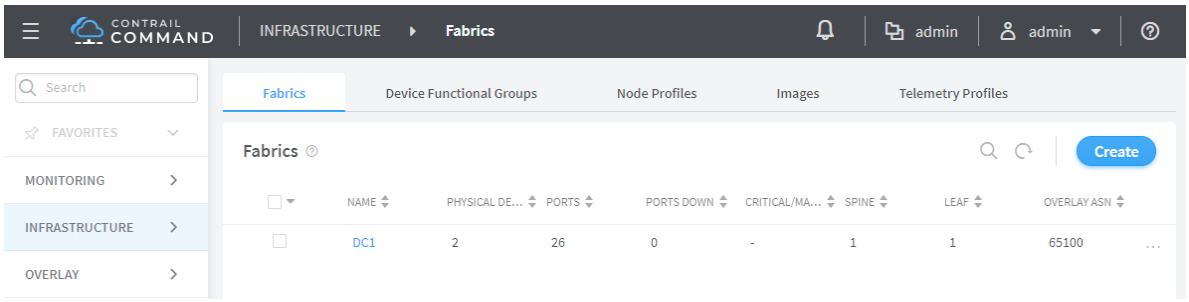


Table 61: Fabrics Page Information

Column	Description
Name	Name of the fabric. **NOTE**: A red alert icon displays next to the fabric name if there are any active critical or major alarms.
Physical Devices	Number of connected physical devices.
Ports	Number of physical ports.
Ports Down	Number of physical ports nonfunctional.
Critical/Major Alarm	Number of critical alarms reported by the telemetry profiles.
Spine	Number of spine devices connected in the fabric.
Leaf	Number of leaf devices connected in the fabric.
Overlay ASN	Autonomous system number for the EVPN overlay.

2. Select the *<Fabric Name>* to view more details about devices, ports, and alarms.

   The Fabric Devices page displays an overview of the fabric devices and their real-time status.

Figure 181: Fabric Devices Page

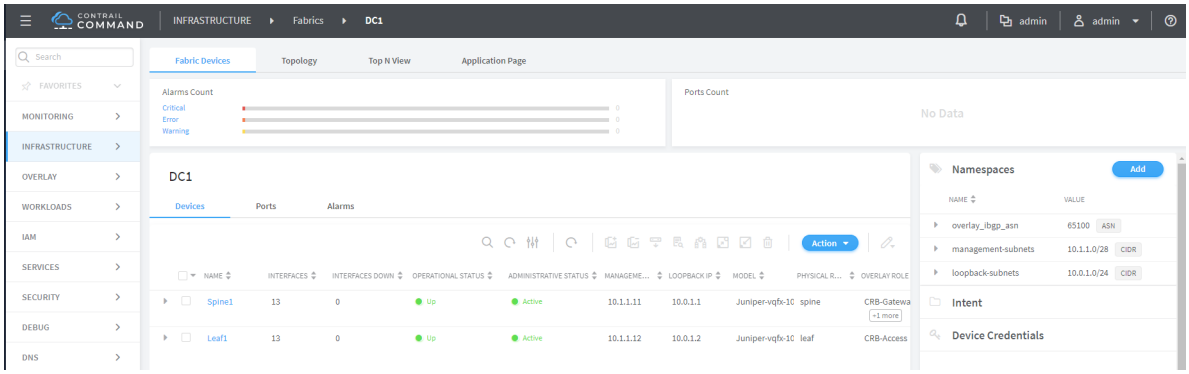


Table 62: Fabric Devices Page Information

Column	Description
Alarms Count	A progress bar chart of critical, error, or warning alarms. Click **Critical**, **Error**, or **Warning** to display only alarms with that severity level.
Ports Count	A pie chart of connected ports showing up or down operational status and administrative status for physical ports.
Name	Name of the fabric device. Click a *<device name>* to navigate to the device's detailed overview page.
Interfaces	Number of connected physical interfaces for that device.
Interfaces Down	Number of physical interfaces nonfunctional for that device.
Operational Status	Device status as Up, Down or At Risk/Changed. This status is based on the telemetry profiles.

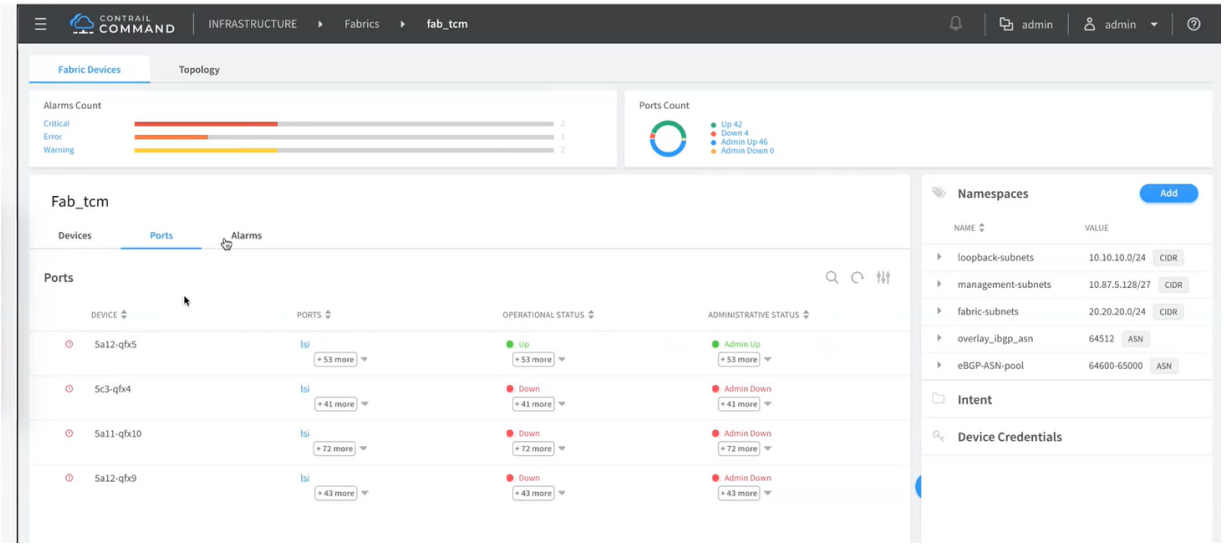
Table 62: Fabric Devices Page Information *(Continued)*

Column	Description
Administrative Status	Device status as Active, Inactive, or Changed. This status applies to the Contrail Networking configuration component versus telemetry profiles.
Management IP	IP address used as the management IP address. Contrail Command uses this IP address to connect to the Contrail Insights Flows node.
Loopback IP	Loopback subnets are used to auto-assign loopback IP addresses to the fabric devices.
Model	Brand and device type.
Physical Role	Device roles define the routing and bridging responsibilities for each device in a fabric. A fabric device can have one physical role and one or more routing bridging roles. Use this page to assign roles to each device in the fabric. See the “Device Roles” section of the Contrail Enterprise Multicloud for Fabric Management guide for information on device roles.
Overlay Role	Routing-bridging role, such as CRBGateway, ERBGateway, and so on.
Namespaces	Refers to the objects, such as BGP ASN pool, Management Subnet, or peer-to-peer (P2P) interface subnet that is assigned to a fabric.
Intent	High level abstract operations that user refers to in terms of the application-to-application connectivity he is looking for. For example VN, logical router, physical network function (PNF), and so on.
Device Credentials	Credentials for accessing fabric devices.

Ports

Click the **Ports** tab to view details about ports.

Figure 182: Fabrics Ports Page Information



The **Ports** tab displays the following information:

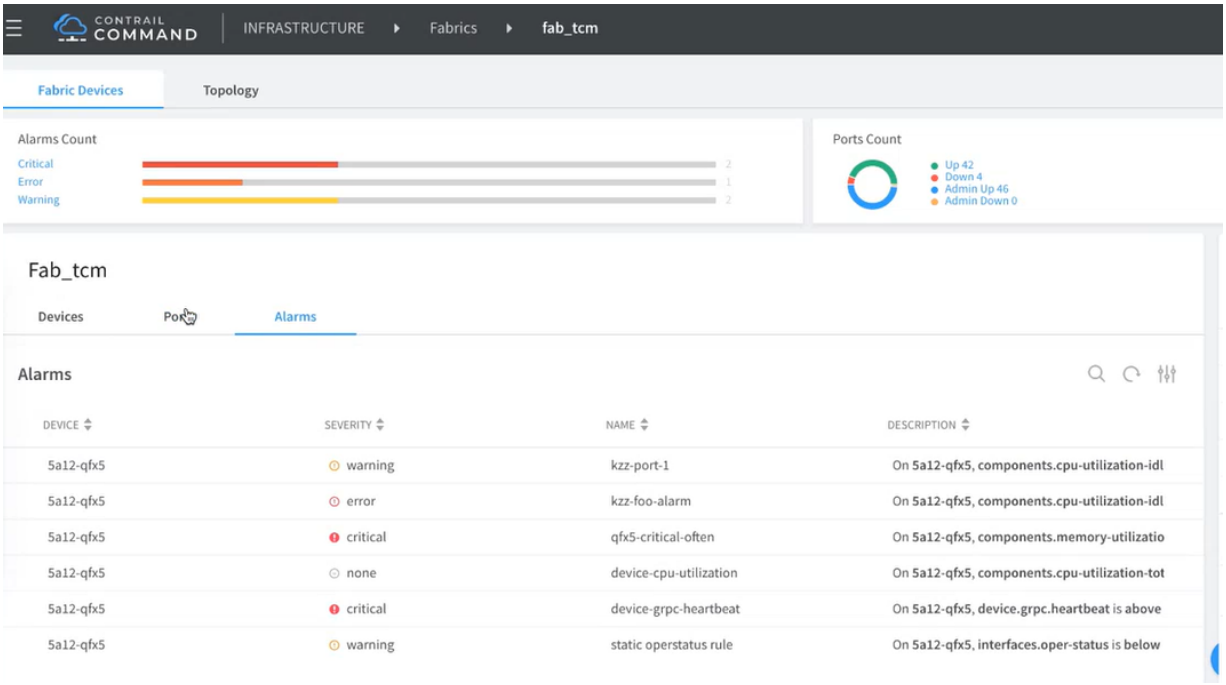
Table 63: Fabrics Ports Page Information

Column	Description
Device	Device name
Ports	Number of ports reported by telemetry profiles.
Operational Status	Device status as Up, Down or At risk/Changed. This is based on information received from the telemetry profiles.
Administrative Status	Device status as Active, Inactive, or Changed. This status applies to the Contrail Networking configuration component versus telemetry profiles.

Alarms

Click the **Alarms** tab to view all alarms on the fabric.

Figure 183: Fabrics Alarm Page



The **Alarms** tab displays the following information:

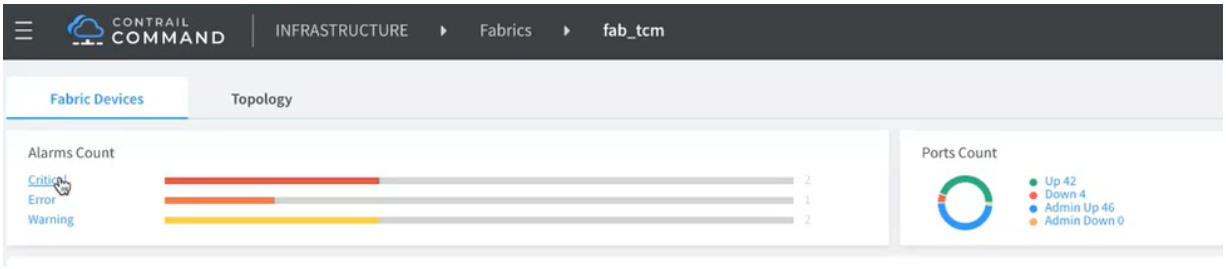
Table 64: Fabrics Alarm Page Information

Column	Description
Device	Device Name
Severity	Alarm severity. Ranging from none to critical. Corresponds with Alarms Count at top of page.
Name	Alarm name.
Description	Brief description of alarm, that is pulled from telemetry profiles.

Alarms Count, Ports Count, and Device Overview

In Alarms Count, click **Critical**, **Error**, or **Warning** to display only alarms with that severity level.

Figure 184: Display Alarms by Severity from Alarms Count



To return to displaying all alarms, click the **Open filters** icon and select the filter values or click **Reset**.

Figure 185: Open Filters Icon in Fabric Alarms

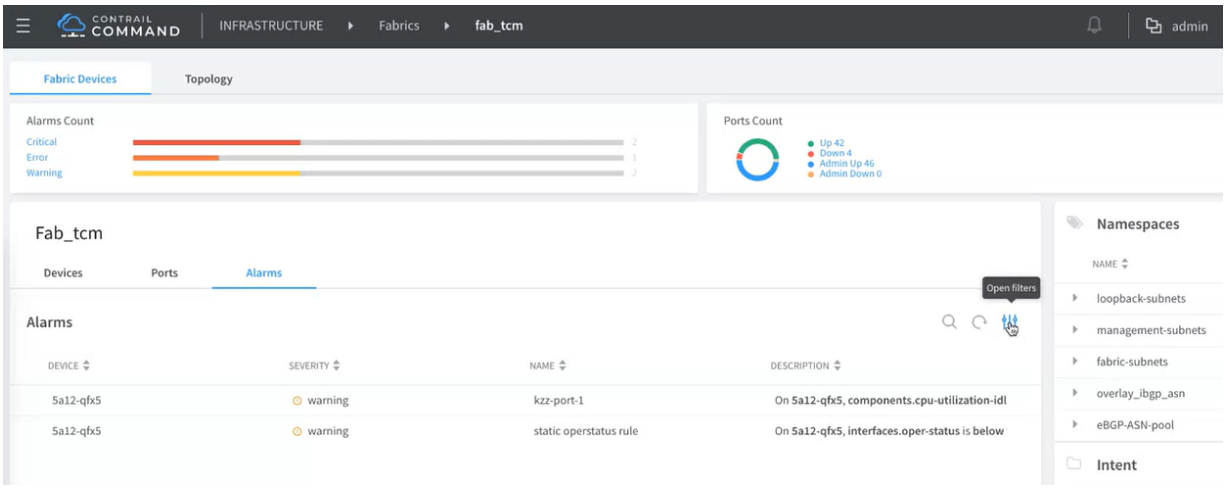
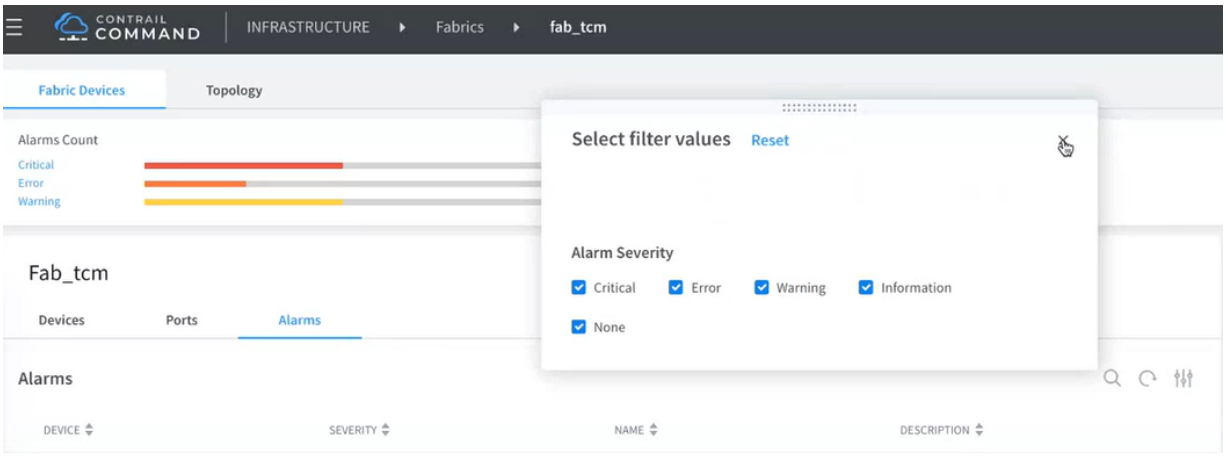


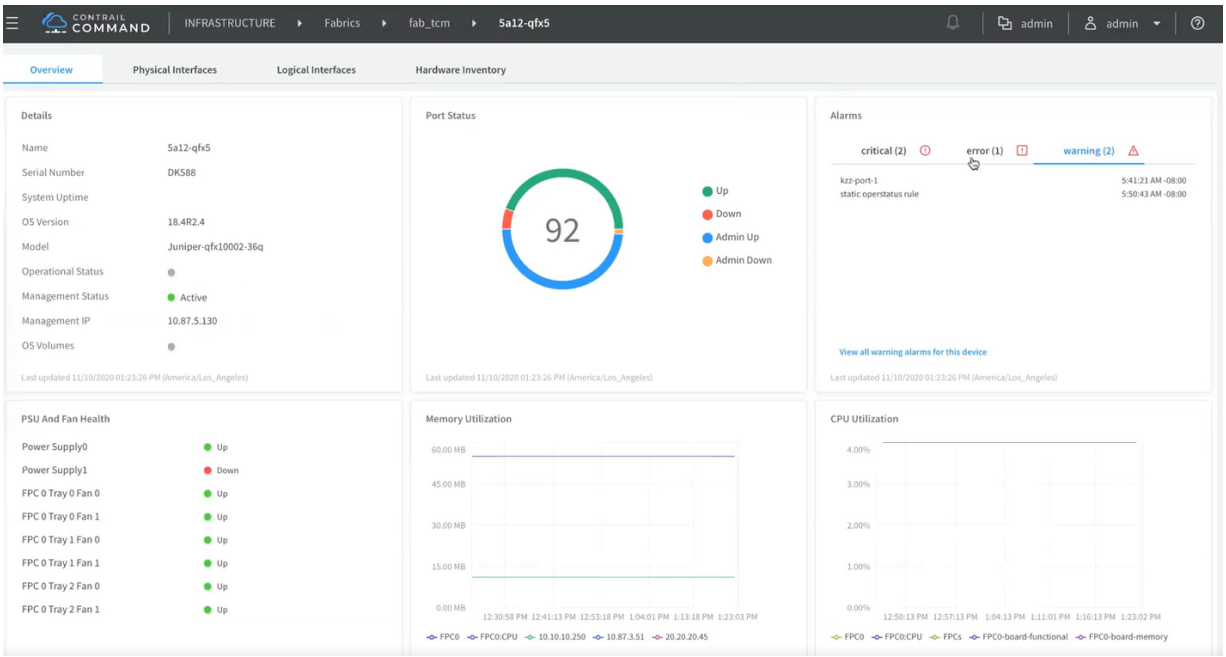
Figure 186: Select Filter Values in Fabric Alarms



NOTE: The same behavior applies to the legend items next to the Ports Count pie chart.

The Device Overview page shows more detail provided by the telemetry profiles. The Alarms panel displays up to eight alarms. Click **View all critical alarms for this device** to see any additional alarms.

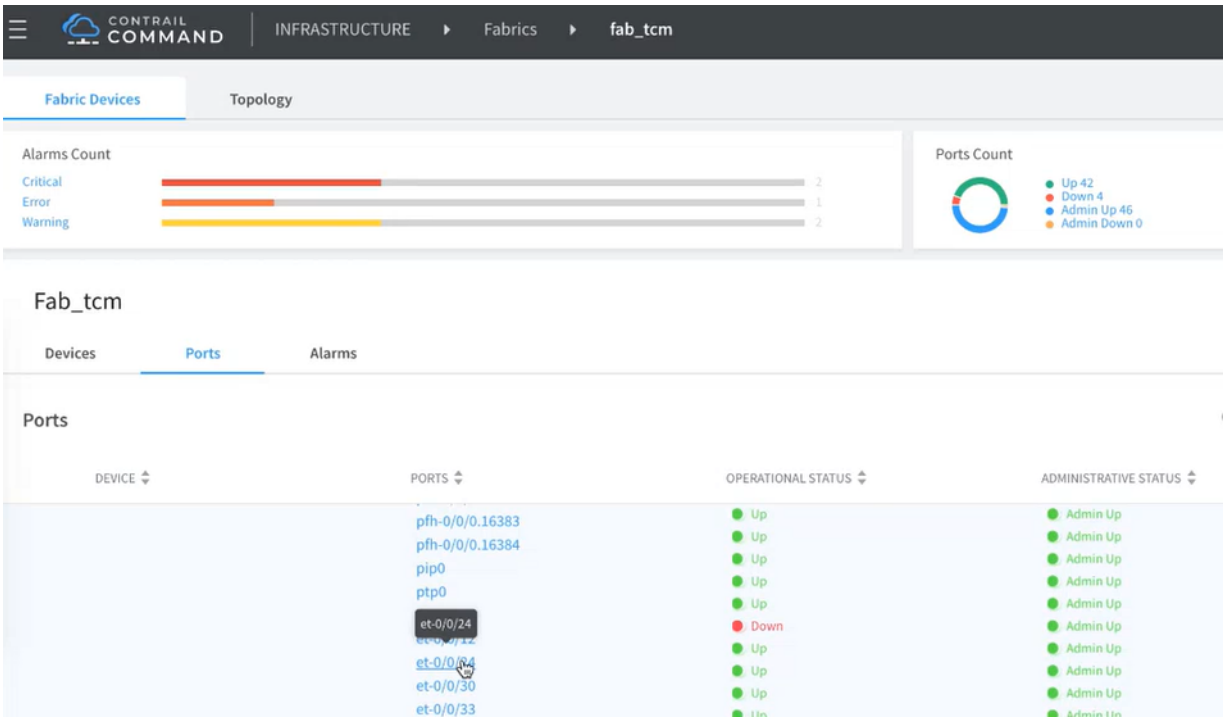
Figure 187: Fabric Device Overview Page



This returns you to the Fabric page and displays device and severity dependent on your choices. Click **Open filters** and select **Reset** to return the page to displaying all devices and alarms. See

In Ports Count, you can click a port name to see more detail. This launches the Port Overview page. Overview charts are drawn specifically for the port that you are viewing. To return to full view of all ports, click the **Open filters** icon and select **Reset.**

Figure 188: Select Fabric Port to View Alarms Specific to Selected Port



Configure and Assign Telemetry Profiles from Fabric Overview

When there are no telemetry metrics configured and assigned to applicable fabric devices, you can accomplish this from the Fabric Overview page. For more information about telemetry profiles, see "Contrail Insights Flows in Contrail Command" on page 339 .

Figure 189: Configure Metrics from Fabric Overview Page

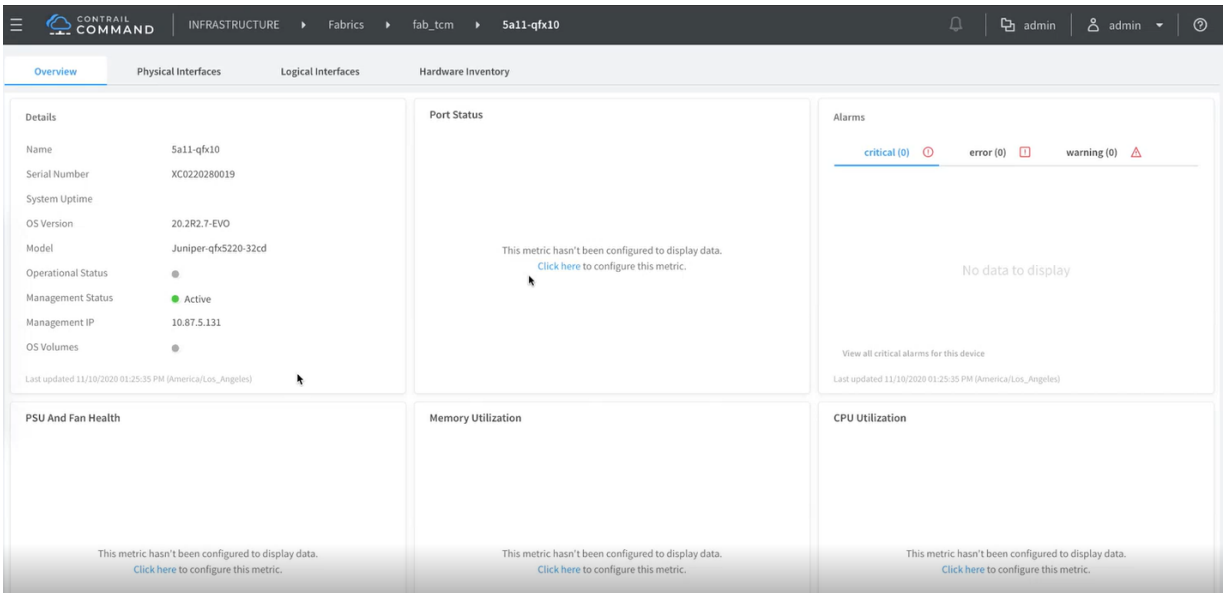


Figure 190: Assign Telemetry Profile Icon

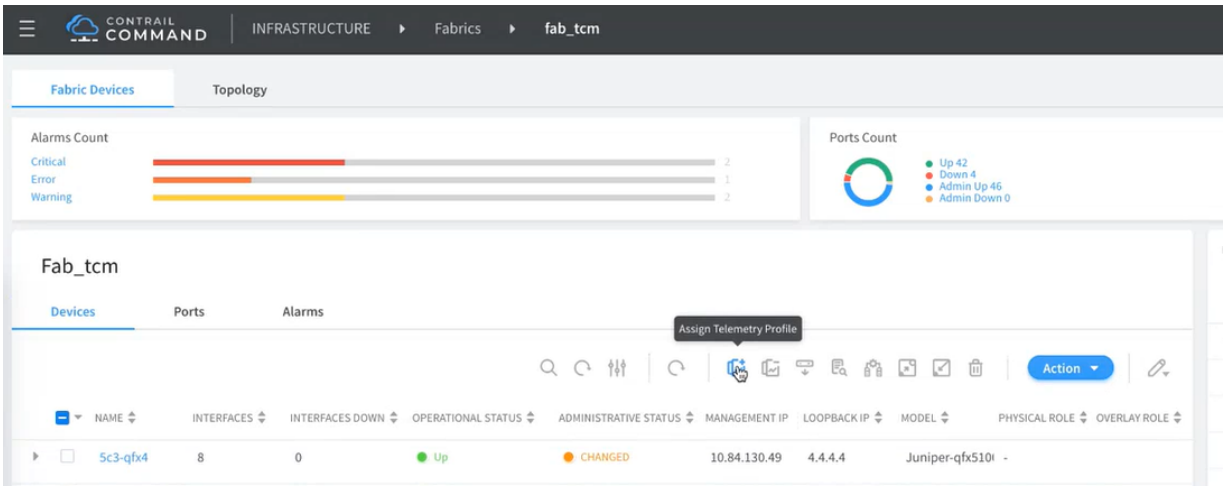
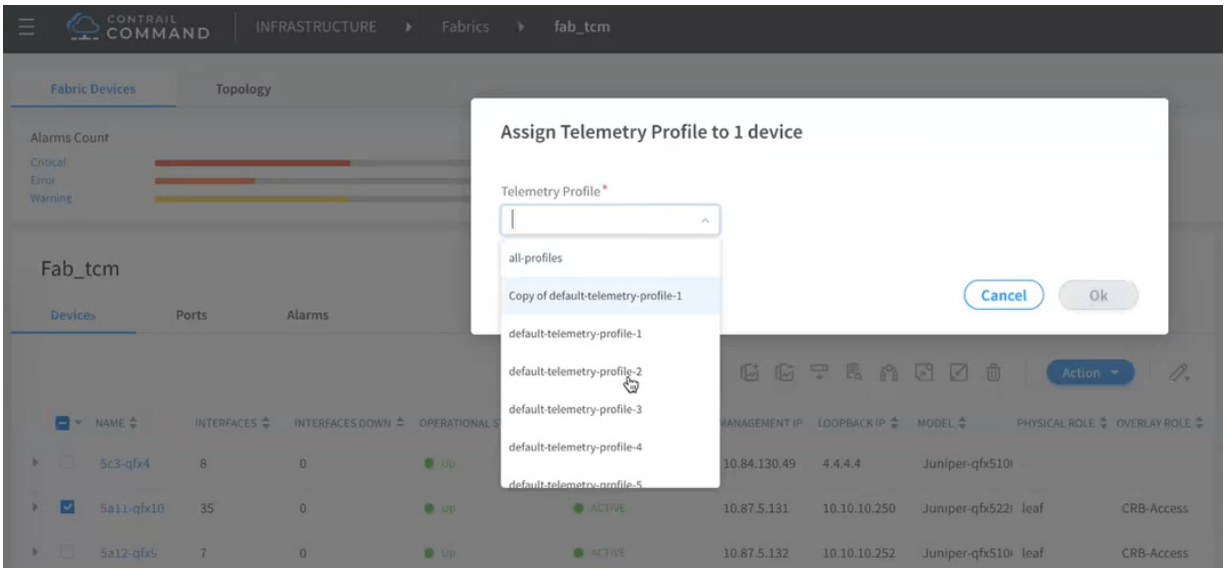


Figure 191: Assign Telemetry Profile from Fabric Overview Page



Change History Table

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

Release	Description
2011	In Contrail Networking Release 2011, the Fabrics page displays additional detail about the health of your devices and interfaces, gathering data from telemetry metrics and profiles configured on these devices. These pages will show if any key performance indicators (KPIs) have crossed a threshold value. You can also configure metrics and assign telemetry profiles from the Fabrics Overview page.

RELATED DOCUMENTATION

Contrail Insights Flows in Contrail Command | 339

Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command | 364

Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command

IN THIS SECTION

- Add Collector Node - No Existing Collector Nodes, No Available Server | 364
- Add Collector Node - No Existing Collector Nodes, Available Server | 368
- Add Collector Node - Existing Collector Nodes, No Available Server | 371
- Add Collector Node - Existing Collector Nodes, Available Server | 373
- Connect Collector Nodes to Fabric | 374
- Edit Existing Collector Nodes | 376
- Remove Collector Nodes from a Contrail Cluster | 376

Contrail Networking Release 2011 supports adding, removing, and reconfiguring collector nodes (also known as *sFlow* nodes) after the system is up and running. Prior to this release, collector nodes could only be added during provisioning.

The provisioning workflow for the collector nodes is:

1. When there are not any existing collector nodes in the deployment, you need to specify the provisioning type (out-of-band or in-band) and the corresponding configuration parameters.
2. After specifying the provisioning type, you create server nodes if none are available or select from the listed server nodes.
3. Next assign the selected server nodes as new collector nodes.

The Add Collector Nodes wizard guides you through the steps, which vary depending on the availability of collector nodes and server nodes. See the following procedures for the add collector nodes workflows.

Add Collector Node - No Existing Collector Nodes, No Available Server

An available server node is one that's not currently assigned as a Contrail Insights sFlow node.

To add an out-of-band collector node when there are not any existing collector nodes and no available server nodes:

1. Select **Infrastructure > Cluster**.

The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays without any collector nodes listed.

3. Click the link **Add Collector Node** or navigate to **Add > Collector Node**.

   The page opens with **Step 1 - Provisioning Type**.

4. Select the Collector Node provisioning type to identify how the Insights Flows node is managed.

   - **Out of Band**: Contrail Insights Flows nodes are managed from an out-of-band management network by default. See Table 65 on page 365 .
   - **In-Band**: Select the in-band option if you want to manage Contrail Insights from an in-band network interface. See Table 66 on page 365 .

Table 65: Provisioning Type: Out of Band Configuration Information

Field	Description
Virtual IP Address	Enter the virtual IP address on the Insights Flows node that connects the node to the management network. The address is entered as a four-octet IP address with no mask; for example, 10.1.1.20.
Show Advanced (Check box)	
Retention Period	Time duration in seconds that you want to keep the collected data. For example, 7200.
Max Retention Bytes	Maximum size of the data to be collected. Default is 0 which indicates unlimited size.

Table 66: Provisioning Type: In-Band Configuration Information

Field	Description
In-Band Collector Configuration	
CIDR	Enter the underlay telemetry infrastructure subnet. The in-band interface on the Contrail Insights Flows node is assigned an IP address from this subnet.

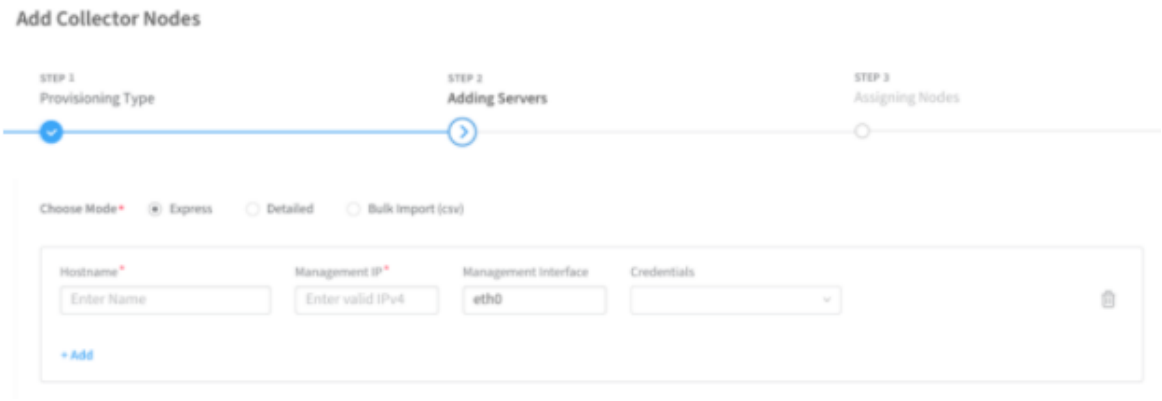
Table 66: Provisioning Type: In-Band Configuration Information *(Continued)*

Field	Description
VLAN ID	Enter the VLAN ID used for the telemetry network.
Management Virtual IP Address	Enter an unused IP address which will be used as the management IP Address. Contrail Command uses this IP address to connect to the Contrail Insights Flows node.
Show Advanced (Check box)	
Retention Period	Time duration in seconds that you want to keep the collected data. For example, 7200.
Max Retention Bytes	Maximum size of the data to be collected. Default is 0 which indicates unlimited size.
AppFormix Flows Configuration Parameters	
Key	Enter a key value for a key value pair on the Contrail Insights Flows server. Key value pairs might need to be entered to use Contrail Insights and Contrail Insights Flows on the same server. See *How to Install Contrail Command and Provision Your Contrail Cluster*. In all other scenarios, key value pairs should only be used by expert users or by users in specialized circumstances.
Value	Enter a value for a key value pair on the Contrail Insights Flows server. Key value pairs might need to be entered to use Contrail Insights and Contrail Insights Flows on the same server. See *How to Install Contrail Command and Provision Your Contrail Cluster*. In all other scenarios, key value pairs should only be used by expert users or by users in specialized circumstances.

5. Click **Next**.

Without a server available, you are directed to **Step 2 - Adding Servers**.

Figure 192: Collector Nodes: Adding Servers



6. Complete the required fields. See Table 67 on page 367 .

Table 67: Collector Nodes: Adding Servers Configuration Information

Field	Description
Management Virtual IP address	Server IP address (for example, 1.1.1.1).
Choose Mode	Options include: *Express*, *Detailed*, or *Bulk Import (CSV)*. We recommend using the *Detailed* or *Bulk Import (CSV)* modes in most environments to ensure all server field data is entered and to avoid performing manual configuration tasks later in the procedure. • *Express*—includes a limited number of required fields to enter for each server or VM. • *Detailed*—provides all fields to enter for each server or VM. • *Bulk Import (CSV)*—Import the physical server or VM fields from a CSV file.
Hostname	Name of the physical server or VM.
Management IP	Management IP address of the physical server or VM.
Management Interface	Name of the management-network facing interface on the physical server or VM.

Table 67: Collector Nodes: Adding Servers Configuration Information *(Continued)*

Field	Description
Credentials	Select any credentials that appear in the drop-down list.
Disk Partition(s)	(Optional) Specify the disk partitions that you want to use. This field is often left blank.
Name	(Network interfaces)—the name of a network-facing interface on the physical server or VM.
IP Address	(Network interfaces)—the IP address of the network-facing interface on the physical server or VM.

7. Click **Next** after completing all fields to add the server or VM.

   The page continues to **Step 3 Assigning Nodes**.

   You can see the added server in the Available Servers box. Listed are the servers that can be provisioned as Contrail Insights nodes. Contrail Insights provides end-to-end visibility into your cloud environment to improve the operations of your network. A Contrail Insights node is needed to run Contrail Insights.

8. Click the **>** icon next to a server to assign it as a Contrail Insights node. The server is moved into the Assigned Contrail Insights Nodes table.

9. Click **Provision**.

   The cluster provisioning begins and the page displays provisioning progress. When the provisioning is completed, you are directed to log in to Contrail Command.

   Continue to "Connect Collector Nodes to Fabric" on page 374 .

Add Collector Node - No Existing Collector Nodes, Available Server

To add an out of band collector node, when there is an available server node but there are no collector nodes:

1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays without any collector nodes listed.

3. Click **Add > Collector Node**.

   The page opens with **Step 1 - Provisioning Type**.

4. Select the Collector Node provisioning type to identify how the Insights Flows node is managed.

   - **Out of Band**: Contrail Insights Flows nodes are managed from an out-of-band management network by default. See Table 68 on page 369 .
   - **In-Band**: Select the in-band option if you want to manage Contrail Insights from an in-band network interface. See Table 69 on page 369 .

Table 68: Provisioning Type: Out of Band Configuration Information

Field	Description
Virtual IP Address	Enter the virtual IP address on the Insights Flows node that connects the node to the management network. The address is entered as a four-octet IP address with no mask; for example, 10.1.1.20.
Show Advanced (Check box)	
Retention Period	Time duration in seconds that you want to keep the collected data. For example, 7200.
Max Retention Bytes	Maximum size of the data to be collected. Default is 0 which indicates unlimited size.

Table 69: Provisioning Type: In-Band Configuration Information

Field	Description
In-Band Collector Configuration	
CIDR	Enter the underlay telemetry infrastructure subnet. The in-band interface on the Contrail Insights Flows node is assigned an IP address from this subnet.

Table 69: Provisioning Type: In-Band Configuration Information *(Continued)*

Field	Description
VLAN ID	Enter the VLAN ID used for the telemetry network.
Management Virtual IP Address	Enter an unused IP address which will be used as the management IP Address. Contrail Command uses this IP address to connect to the Contrail Insights Flows node.
Show Advanced (Check box)	
Retention Period	Time duration in seconds that you want to keep the collected data. For example, 7200.
Max Retention Bytes	Maximum size of the data to be collected. Default is 0 which indicates unlimited size.
AppFormix Flows Configuration Parameters	
Key	Enter a key value for a key value pair on the Contrail Insights Flows server. Key value pairs might need to be entered to use Contrail Insights and Contrail Insights Flows on the same server. See *How to Install Contrail Command and Provision Your Contrail Cluster*. In all other scenarios, key value pairs should only be used by expert users or by users in specialized circumstances.
Value	Enter a value for a key value pair on the Contrail Insights Flows server. Key value pairs might need to be entered to use Contrail Insights and Contrail Insights Flows on the same server. See *How to Install Contrail Command and Provision Your Contrail Cluster*. In all other scenarios, key value pairs should only be used by expert users or by users in specialized circumstances.

5. Click **Next**.

   The page continues to **Step 2 - Assigning Nodes**.

   The Available Servers table lists the servers that can be provisioned as Contrail Insights nodes. Contrail Insights provides end-to-end visibility into your cloud environment to improve the operations of your network. A Contrail Insights node is needed to run Contrail Insights.

6. Click the **>** icon next to a server to assign it as a Contrail Insights node. The server is moved into the Assigned Contrail Insights Nodes table.

7. Click **Provision**.

   The cluster provisioning begins and the page displays provisioning progress. When the provisioning is completed, you are directed to log in to Contrail Command.

   Continue to "Connect Collector Nodes to Fabric" on page 374 .

Add Collector Node - Existing Collector Nodes, No Available Server

When there are existing collector nodes in the deployment, you are not required to specify the provisioning type.

To add a collector node when there are existing collector nodes and no available server nodes:

1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays with collector nodes listed.

3. Click **Add > Collector Node**.

   The page opens with **Step 1 Adding Servers**.

4. Complete the required fields. See Table 70 on page 371

   Table 70: Collector Nodes: Adding Servers Configuration Information

Field	Description
Management Virtual IP address	Server IP address (for example, 1.1.1.1).

Table 70: Collector Nodes: Adding Servers Configuration Information *(Continued)*

Field	Description
Choose Mode	Options include: *Express*, *Detailed*, or *Bulk Import (CSV)*. We recommend using the *Detailed* or *Bulk Import (CSV)* modes in most environments to ensure all server field data is entered and to avoid performing manual configuration tasks later in the procedure. • *Express*—includes a limited number of required fields to enter for each server or VM. • *Detailed*—provides all fields to enter for each server or VM. • *Bulk Import (CSV)*—Import the physical server or VM fields from a CSV file.
Hostname	Name of the physical server or VM.
Management IP	Management IP address of the physical server or VM.
Management Interface	Name of the management-network facing interface on the physical server or VM.
Credentials	Select any credentials that appear in the drop-down list.
Disk Partition(s)	(Optional) Specify the disk partitions that you want to use. This field is often left blank.
Name	(Network interfaces)—the name of a network-facing interface on the physical server or VM.
IP Address	(Network interfaces)—the IP address of the network-facing interface on the physical server or VM.

5. Click **Next** after completing all fields to add the server or VM.

   The page continues to **Step 2 Assigning Nodes**.

   You can see the added server in the Available Servers box. Listed are the servers that can be provisioned as Contrail Insights nodes. Contrail Insights provides end-to-end visibility into your cloud environment to improve the operations of your network. A Contrail Insights node is needed to run Contrail Insights.

6. Click the **>** icon next to a server to assign it as a Contrail Insights node. The server is moved into the Assigned Contrail Insights Nodes table. If you are assigning an in-band server, you are prompted to enter the in-band interface.

7. Click **Provision**.

   The cluster provisioning begins and the page displays provisioning progress. When the provisioning is completed, you are directed to log in to Contrail Command.

   Continue to "Connect Collector Nodes to Fabric" on page 374 .

Add Collector Node - Existing Collector Nodes, Available Server

To add a collector node when there are existing collector nodes and an available server node:

1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

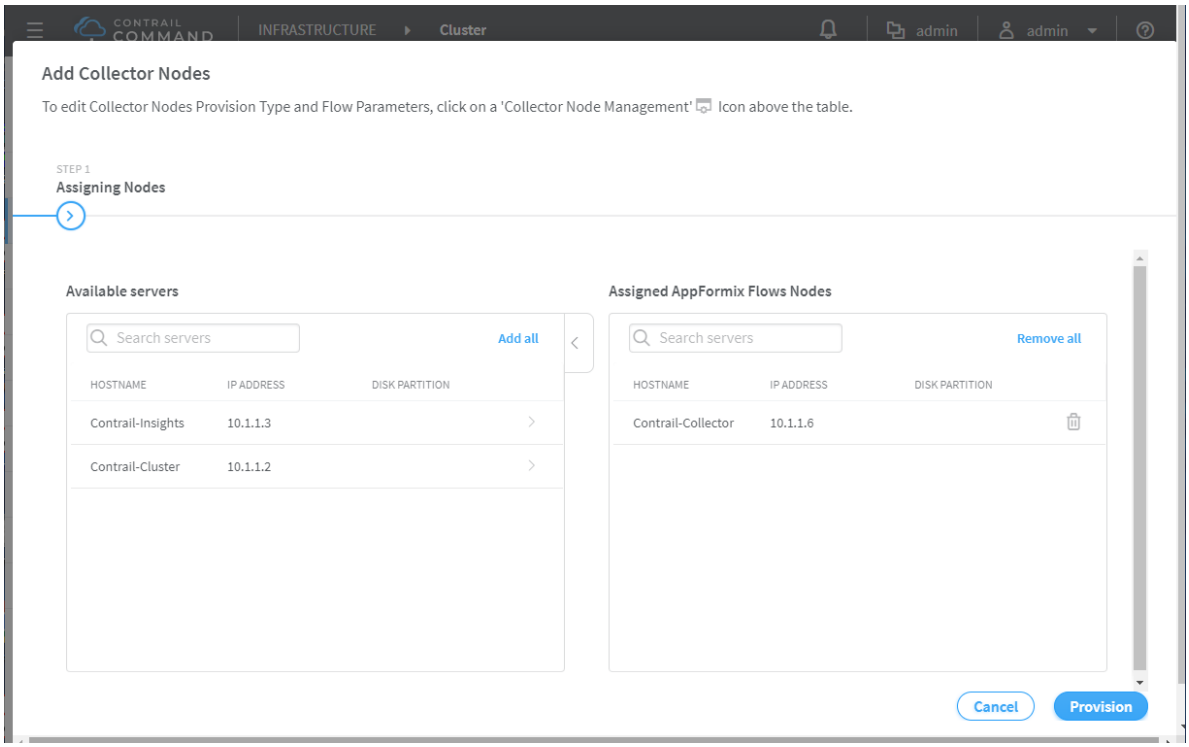
   The page displays the existing collector nodes.

3. Click **Add > Collector Nodes**.

   Add Collector Nodes page with **Step 1 - Assigning Nodes**.

4. Click the **>** icon next to a server to assign it as a Contrail Insights node. The server is moved into the Assigned Contrail Insights Nodes table. If you are assigning an in-band server, you are prompted to enter the in-band interface.

Figure 193: Collector Nodes: Assigning Nodes



5. Click **Provision**.

   The cluster provisioning begins and the page displays provisioning progress. When the provisioning is completed, you are directed to log in to Contrail Command.

   Continue to "Connect Collector Nodes to Fabric" on page 374 .

Connect Collector Nodes to Fabric

To connect collector nodes to a fabric:

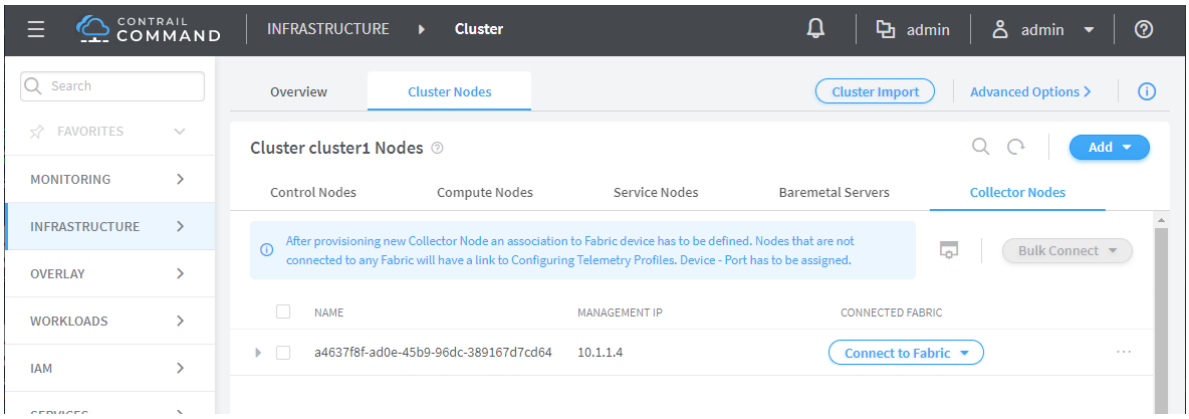
1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays with a message stating the new collector nodes need to be associated to a fabric device. In the Connected Fabric column, if the collector node is connected to a fabric the fabric name displays. When a collector node is not connected, the **Connect to Fabric** drop-down list shows the unconnected nodes. Two options to connect are offered:

Figure 194: Collector Nodes: Connect to Fabric



Two options to connect are offered:

- Click **Connect to Fabric** to connect a singular collector node to a fabric device.
- Click **Bulk Connect** to connect multiple selected collector nodes to a fabric device.

3. Click **Connect to Fabric**, then select the unconnected node from the drop-down list.
4. Complete the required fields. See Table 71 on page 375 .

Table 71: Collector Nodes: In-Band Collector Configuration

Field	Description
In-Band Collector Configuration	
CIDR	Enter the underlay telemetry infrastructure subnet. The in-band interface on the Contrail Insights Flows node is assigned an IP address from this subnet.
Virtual IP Address	Enter the virtual IP address on the Insights Flows node that connects the node to the management network. The address is entered as a four-octet IP address with no mask; for example, 10.1.1.20.
In-Band Collector Details	All in-band configured collector nodes will display here.

5. Click **Next** after completing all fields to connect to a fabric.

The page continues to the **Assign Telemetry Profiles** step.

See "Configuring Contrail Insights Flows by Assigning Telemetry and sFlow Profiles to Devices" on page 341 .

Edit Existing Collector Nodes

To edit existing collector nodes:

1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

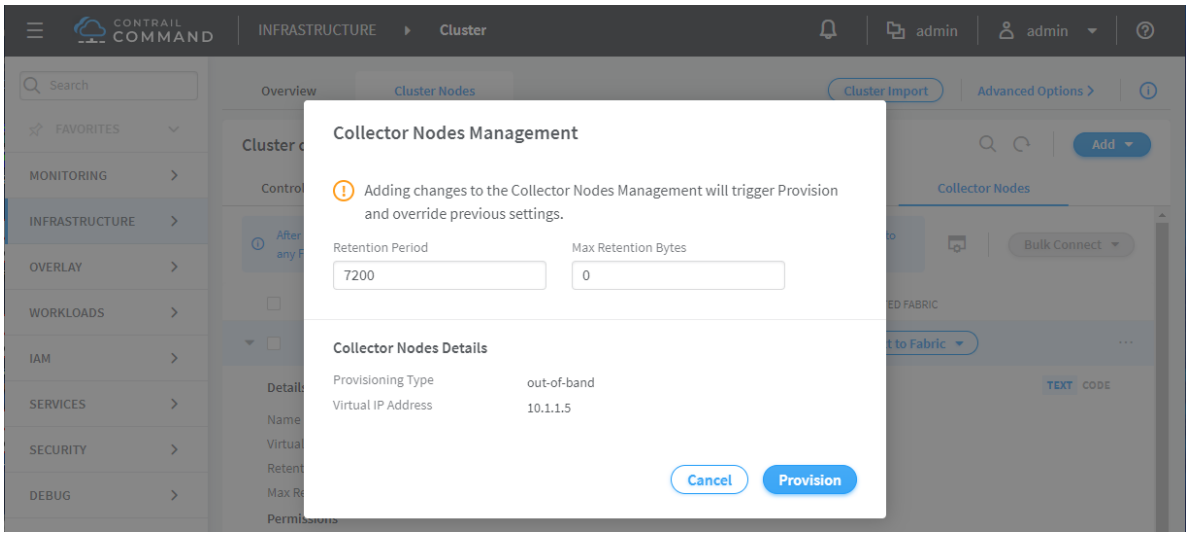
2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays the existing flow collector nodes.

3. Click the **Collector Nodes Management** icon.

   The Collector Nodes Management dialog box displays details about provisioning type and flow parameters.

Figure 195: Collector Nodes: Collector Nodes Management



4. Complete the edits and click **Provision**.

   The cluster provisioning begins and the page displays provisioning progress. When the provisioning is completed, you are directed to log in to Contrail Command.

Remove Collector Nodes from a Contrail Cluster

To delete an existing collector node:

1. Select **Infrastructure > Cluster**.

   The Cluster Overview page displays.

2. Click the **Collector Nodes** tab.

   The Collector Nodes page displays the existing flow collector nodes.

3. Click the trash can icon in the row for the flow collector node you want to remove.

4. Click **Delete**.

   A dialog box displays asking you to confirm.

   If your deletion takes the sFlow project below the thresholds, then you are alerted that this will enable sFlow to become unstable. To continue removing the collector node, select the **Unsafe Delete** check box.

5. Click **Delete**.

   The cluster provisioning begins and the page displays provisioning progress.

6. Click **Proceed to login**.

   The log in dialog box appears.

7. Log in to Contrail Command.

   The Collector Nodes page displays and confirms the node is deleted.

Change History Table

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

Release	Description
2011	Contrail Networking Release 2011 supports adding, removing, and reconfiguring collector nodes (also known as *sFlow* nodes) after the system is up and running. Prior to this release, collector nodes could only be added during provisioning.

RELATED DOCUMENTATION

Contrail Insights Flows in Contrail Command | 339

Viewing Telemetry KPI Alarms for Fabric Devices and Ports | 353

Adding or Deleting sFlow Collector Nodes by Modifying instances.yml

IN THIS SECTION

- Caveats | 380

Contrail Networking Release 2011 supports adding, removing, and reconfiguring collector nodes (also known as *sFlow* nodes) after the system is up and running. Prior to this release, collector nodes could only be added during provisioning.

This topic describes adding or deleting flow collector nodes to an existing Contrail cluster by modifying the `instances.yml` file, which gets generated when Contrail cluster is provisioned by using Contrail Command UI. The `instances.yml` file is created inside `/var/tmp/contrail_cluster/<contrail_cluster_id>` in `contrail_command` container.

You can:

- Add flow collector nodes to a Contrail cluster without any existing flow collector node.
- Add flow collector nodes to a Contrail cluster with existing flow collector nodes.
- Delete flow collector nodes from a Contrail cluster.

When a new flow collector node is added, or an existing flow collector node is deleted by modifying the existing `instances.yml` file:

1. Ansible playbook will run to first cleanup the flow collector nodes which need to be deleted.
2. Then Ansible playbook will run to set up the new flow collector nodes and to rebuild the cluster.

To add flow collector nodes:

1. Modify the `instances.yml` file to add flow-collector nodes.

   Existing cluster details are in the `instances.yml`. The `appformix_flows` role designates Contrail Insights Flows node. The following example shows a snippet of section for `instances.yml` for existing Contrail Insights Flows node details:

```
instances:
  host1:
   ip: 10.87.3.85
   provider: bms
```

```
    roles:
     appformix_bare_host:
     appformix_flows:
```

New snippet for `instances.yml` to register one more node, `host2`, in the Contrail Insights Flows cluster:

```
instances:
  host1:
   ip: 10.87.3.85
   provider: bms
   roles:
    appformix_flows:
  host2:
   ip: 10.87.3.86
   provider: bms
   roles:
```

2. Run the following commands to add flow collector nodes:

```
cd /usr/share/contrail/appformix-ansible-deployer/xflow/
. venv/bin/activate
bash deploy_insights_flows.sh <instance.yml file> --cluster-id <contrail_cluster_id>
```

To delete nodes from the Contrail cluster:

1. Modify the `instances.yml` file to delete flow collector nodes. The following example shows a snippet of section for `instances.yml` for existing Contrail Insights Flows node details:

```
instances:
  host1:
   ip: 10.87.3.85
   provider: bms
   roles:
    appformix_bare_host:
    appformix_flows:
  host2:
   ip: 10.87.3.86
   provider: bms
   roles:
```

```
    appformix_bare_host:
    appformix_flows:
```

New snippet for `instances.yml` to deregister `host2` from the Contrail Insights Flows cluster by detaching the `appformix_flows` role from `host2`:

```
instances:
  host1:
   ip: 10.87.3.85
   provider: bms
   roles:
    appformix_bare_host:
    appformix_flows:
  host2:
   ip: 10.87.3.86
   provider: bms
   roles:
    appformix_bare_host:
```

2. Run the following commands to delete the flow collector nodes:

```
cd /usr/share/contrail/appformix-ansible-deployer/xflow/
. venv/bin/activate
bash deploy_insights_flows.sh <instance.yml file> -cluster-id <contrail_cluster_id>
```

Caveats

- You can provision the collectors only once before fabric onboarding.
- You cannot add new Contrail Insights Flows nodes to the cluster after initial provisioning.
- Currently, only sFlow targets are supported.
- Contrail Insights Flows nodes can be connected to only one leaf.

Change History Table

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

Release	Description
2011	Contrail Networking Release 2011 supports adding, removing, and reconfiguring collector nodes (also known as *sFlow* nodes) after the system is up and running. Prior to this release, collector nodes could only be added during provisioning.

RELATED DOCUMENTATION

Adding, Editing, and Deleting sFlow Collector Nodes in Contrail Command | 364

Contrail Insights Flows in Contrail Command | 339

Configuring Contrail Insights Alarms using Contrail Command

IN THIS SECTION

- Contrail Insights Alarms Overview | 382
- Contrail Insights Alarms Operation | 383
- Alarm Definition | 387
- Configuring an Alarm Rule | 403

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

Static User-provided static threshold is used for comparison.

Dynamic Dynamically-learned adaptive threshold is used for comparison.

NOTE: For Contrail Insights releases prior to 3.2.6: In order to configure alarms, your Contrail Insights license subscription must be active.

Contrail Insights Alarms Overview

For both static and dynamic alarms, Contrail Insights Agent continuously collects measurements of metrics (see Metrics Collected by Contrail Insights) 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 Contrail Insights 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, Contrail Insights 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, Contrail Insights 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.

Contrail Insights Alarms Operation

Contrail Insights 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 Contrail Insights 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

Contrail Insights 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 196: Alarm Generation Mechanics

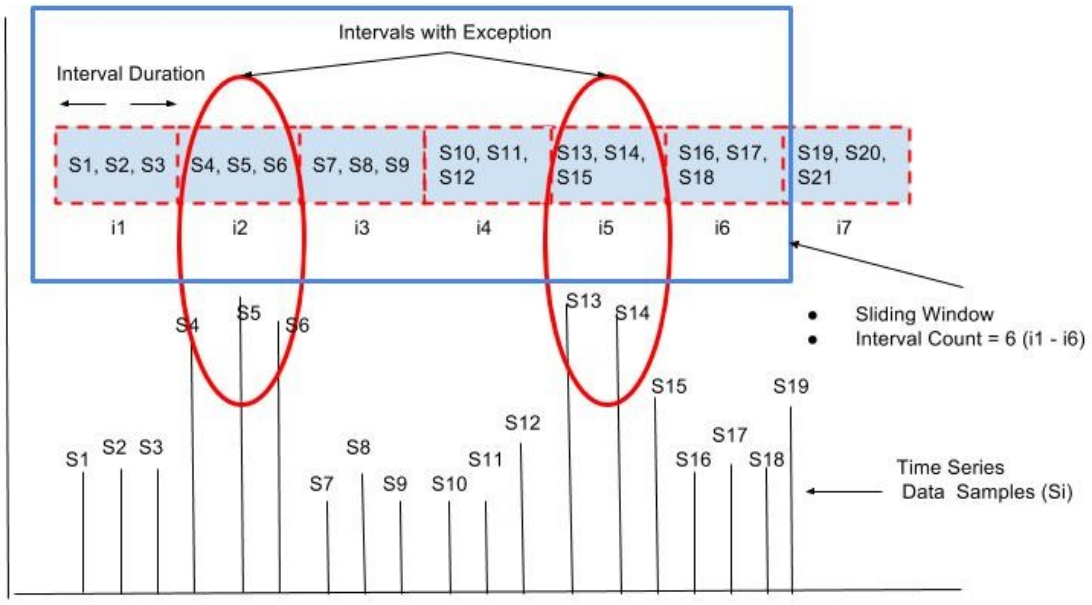


Figure 196 on page 384 shows an example in which the sliding window consists of six adjacent measurement intervals (i1 to i6), as specified by the Interval Count parameter. In measurement interval i1, the average of samples S1, S2, S3 is computed as S_{avg}. Depending on the alarm type *static* or *dynamic*, 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).

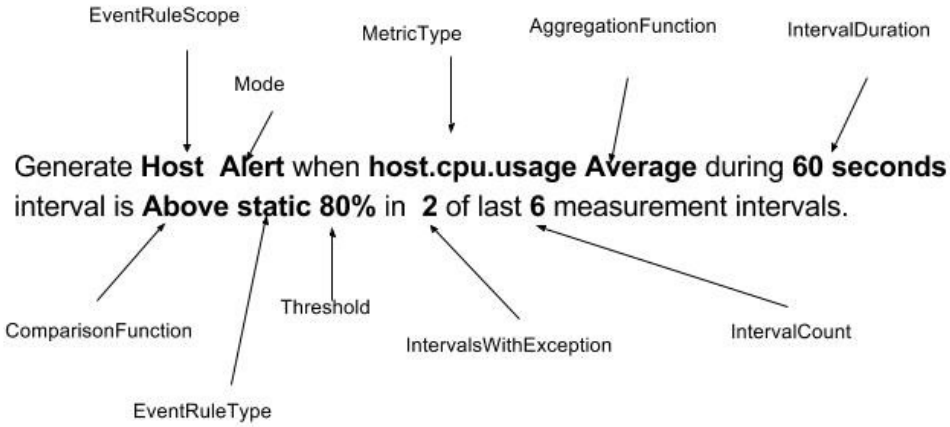
In the example in Figure 196 on page 384 , the agent determines that two intervals, i2 and i5, are *intervals with exception* by comparing the aggregate value for the measurement interval with a static threshold or dynamic baseline, depending on alarm type. Assuming interval i1 is the first interval for which the alarm is configured, the alarm becomes active at end of interval i6, when Contrail Insights 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 197 on page 385 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 197 on page 385 identifies the components in a static alarm definition.

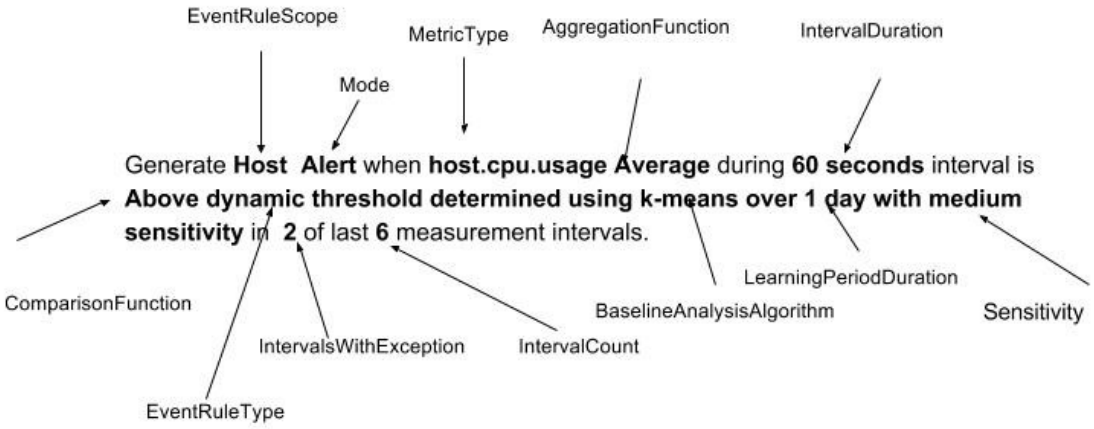
Figure 197: Static Alarm Definition



Dynamic Alarm

A dynamic alarm threshold is learned by Contrail Insights using historical data for the set of entities for which an alarm is configured. Figure 198 on page 386 shows an example of a dynamic alarm definition and identifies the components in a dynamic alarm definition.

Figure 198: 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 Contrail Insights 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, Contrail Insights 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. Contrail Insights Agent compares real-time measurements to the range defined by:

```
mean - sensitivity * std_dev < x < mean + sensitivity * std_dev
```

Alarm Definition

Figure 197 on page 385 shows an example of a static alarm definition. Every alarm definition has the following components shown in Table 72 on page 388 .

Table 72: 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	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. • 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.

Table 72: Alarm Definition Components *(Continued)*

Item	Options	Description
Service	RabbitMQ, MySQL, Ceph, OpenStack, Cassandra, Contrail, ScaleIO	When selected, you can configure alarms for RabbitMQ, MySQL, Ceph, OpenStack, Cassandra, Contrail, and ScaleIO services.
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 *Value* Is *Above* Threshold *-8*. Italics in example represent variables.

Table 72: Alarm Definition Components *(Continued)*

Item	Options	Description
Threshold	Threshold value	A numeric value to which measurements are compared. Contrail Insights supports two types of thresholds: static or dynamic. • 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 201 on page 402 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. The required parameters for dynamic threshold are:

Table 72: Alarm Definition Components *(Continued)*

Item	Options	Description
		• Baseline Analysis Algorithm • Learning Period Duration • Sensitivity Table 73 on page 394 describes the required parameters for a dynamic alarm and the supported options.
Baseline Analysis Algorithm	k-means, ewma	Table 73 on page 394 describes these options. See Figure 201 on page 402 and Figure 202 on page 403 for baseline analysis examples
Learning Period Duration	1 week, 1 month	Table 73 on page 394 describes these options.
Sensitivity	Low, medium, high	Table 73 on page 394 describes these options.
Severity	None, information, warning, error, critical	Indicates seriousness of the alarm. *Critical* indicates a major alarm. *Information* 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.

Table 72: Alarm Definition Components *(Continued)*

Item	Options	Description
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 76 on page 397 lists and describes the aggregation functions for alarm processing.
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 77 on page 398 shows different comparison functions supported for Contrail Insights alarms. Figure 199 on page 400 and Figure 200 on page 401 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. *Critical* indicates a major alarm. *Information* indicates a minor alarm.
Notification	None, PagerDuty, Custom Service, Service Now, Slack	Methods of notification alerting you to conditions of operation.

Table 72: Alarm Definition Components *(Continued)*

Item	Options	Description
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 198 on page 386 , 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.
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 198 on page 386 , 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 73 on page 394 describes the required parameters for a dynamic alarm and the supported options.

Table 73: 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.	• K-Means clustering • Exponential Weighted Mean Average (EWMA)
Learning Period Duration	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. 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, Contrail Insights Agent evaluates metrics as soon as enough data is present to learn the first set of baselines. Example: When Learning Duration is 1 day, the agent compares metrics to per-hour baselines for the last 24 hours. Example: When Learning Duration is 1 week, the agent compares metrics to per-hour baselines for the last 7 x 24 hours.	• 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.

Table 73: Required Parameters for Dynamic Alarm *(Continued)*

Required Parameters for Dynamic Threshold	Description	Supported Options
Sensitivity	The dynamic baseline provides a normal operating region of a given metric for a given scope. As seen in Figure 201 on page 402 , the dynamic baseline is a tuple which has mean and std-dev applicable for a specific hour of the day. 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 *low* 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.	• 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 74 on page 395 shows all possible states for an alarm with the mode configured as alert.

Table 74: 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.

Table 74: States for Alarm Mode Defined as Alert *(Continued)*

State	Description
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 75 on page 396 shows all possible states for an alarm with the mode configured as event.

Table 75: 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 *triggered*. 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 76 on page 397 lists and describes the aggregation functions for alarm processing.

Table 76: Aggregation Functions for Alarm Processing

Aggregation Function	Description
Average	Statistical average of all data samples received within one measurement interval. Example: Generate **Host Alert** when **Cpu-Usage Average** during a **60 seconds** interval is **Above 80%** of **2** of the last **3** measurement intervals. 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.
Sum	Sum of all data samples received within one measurement interval. Example: Generate **Host Alert** when **Cpu-Usage Sum** during a **60 seconds** interval is **Above 250%** of **2** of the last **3** measurement intervals. 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.
Max	Maximum sample value observed within one measurement interval. Example: Generate **Host Alert** when **Cpu-Usage Max** during a **60 seconds** interval is **Above 95%** of **2** of the last **3** measurement intervals. 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.
Min	Minimum sample value observed within one measurement interval. Example: Generate **Host Alert** when **Cpu-Usage Min** during a **60 seconds** interval is **Below 5%** of **2** of the last **3** measurement intervals. 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.

Table 76: Aggregation Functions for Alarm Processing *(Continued)*

Aggregation Function	Description
Std-Dev	Standard Deviation of the time-series data is determined based on the samples received until current measurement interval. 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. 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.

Comparison Functions for Alarm Processing

Figure 199 on page 400 and Figure 200 on page 401 show examples of the Comparison Function, showing both increases and decreases at a minimum rate.

Table 77 on page 398 shows different comparison functions supported for Contrail Insights alarms.

Table 77: Comparison Functions for Alarm Processing

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

Table 77: Comparison Functions for Alarm Processing *(Continued)*

Comparison Operator	Description
Increasing-at-a-minimum-rate-of	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 199 on page 400 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. Example: Generate **Host Alert** when the host.network.ingress.bit_rate **average** during a **60 seconds** interval is **increasing-at-a-minimum-rate-of 25%** of **2** of the last **3** measurement intervals. 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.
Decreasing-at-a-minimum-rate-of	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 200 on page 401 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. Example: Generate **Host Alert** when the host.network.egress.bit_rate **average** during a **60 seconds** interval is **decreasing-at-a-minimum-rate-of 25%** of **2** of the last **3** measurement intervals. 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.

Figure 199: Comparison Function Showing Increasing-at-a-minimum-rate-of

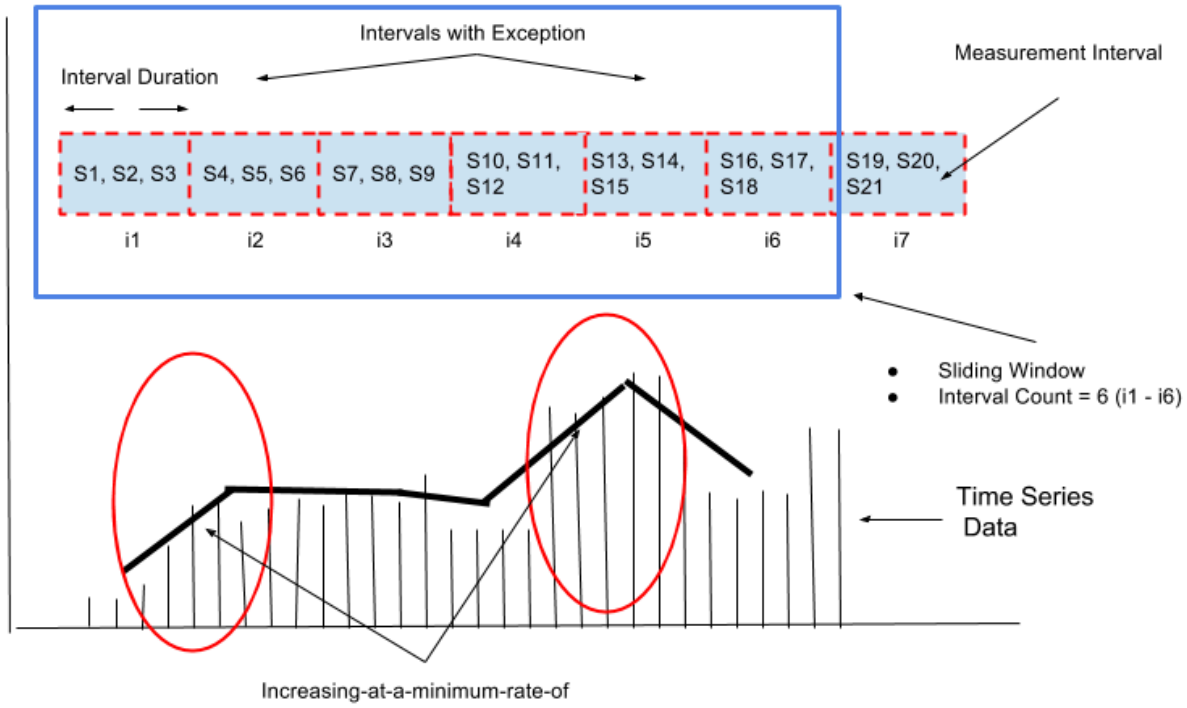
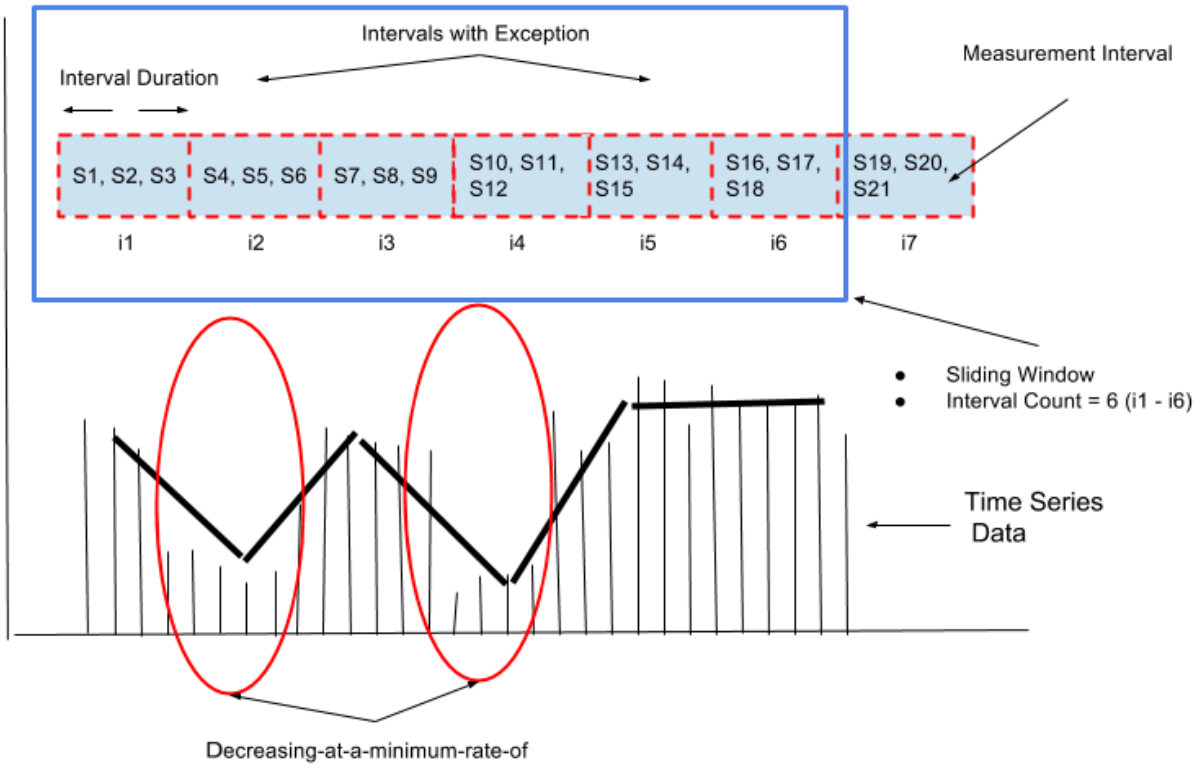


Figure 200: Comparison Function Showing Decreasing-at-a-minimum-rate-of



Dynamic Baseline Examples

Figure 201 on page 402 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 201: Dynamic Baseline Determined by Last 24 Hours of Data and K-Means Clustering Algorithm

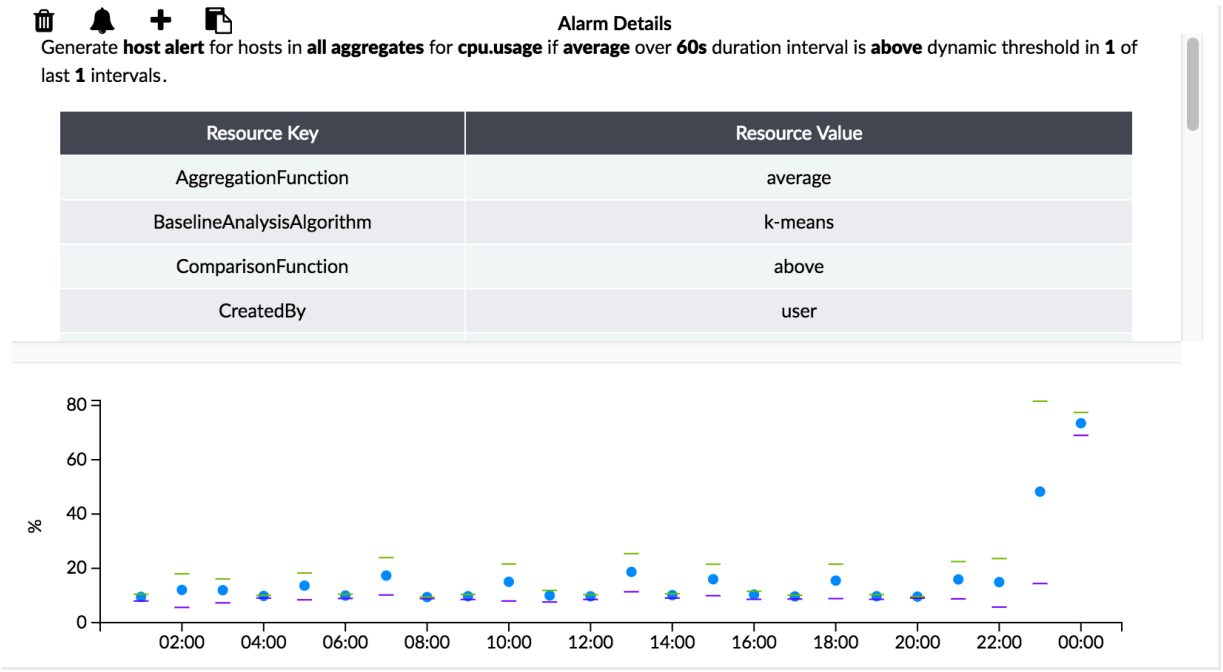
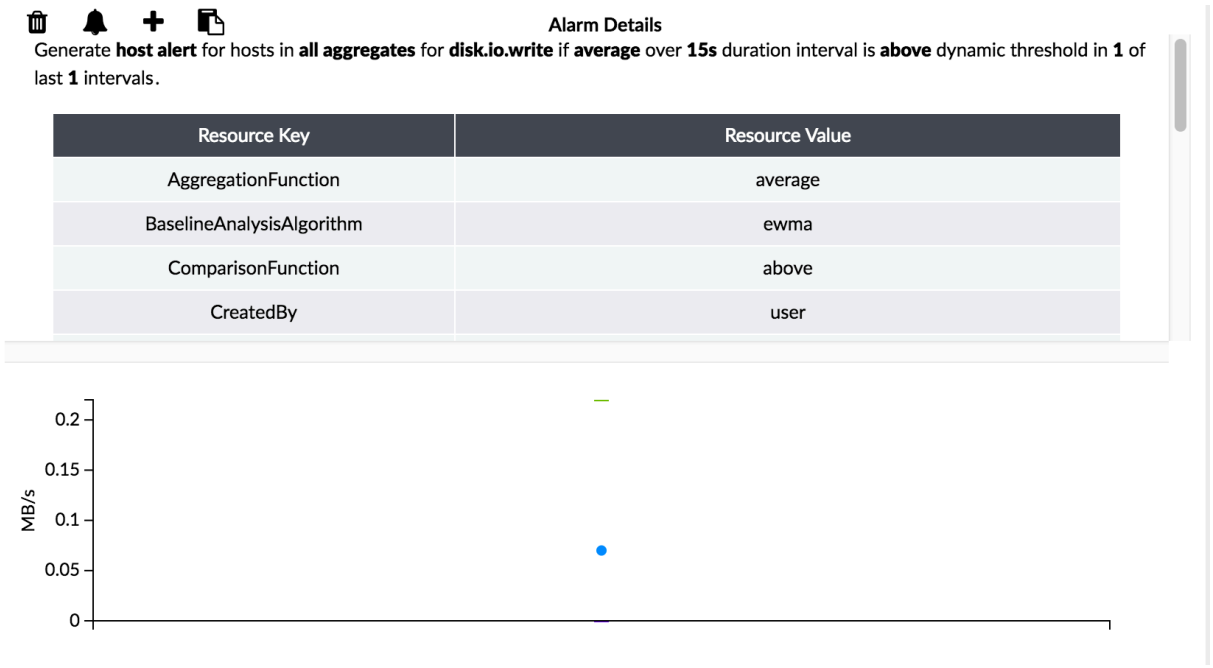


Figure 202 on page 403 shows the dynamic baseline computed by 24 hours of historical data using the EWMA algorithm. This baseline is used for the next 1 hour for alarm generation until it is updated again using the most recent 24 hours of data.

Figure 202: Dynamic Baseline Determined by Last 24 Hours of Historical Data Using EWMA

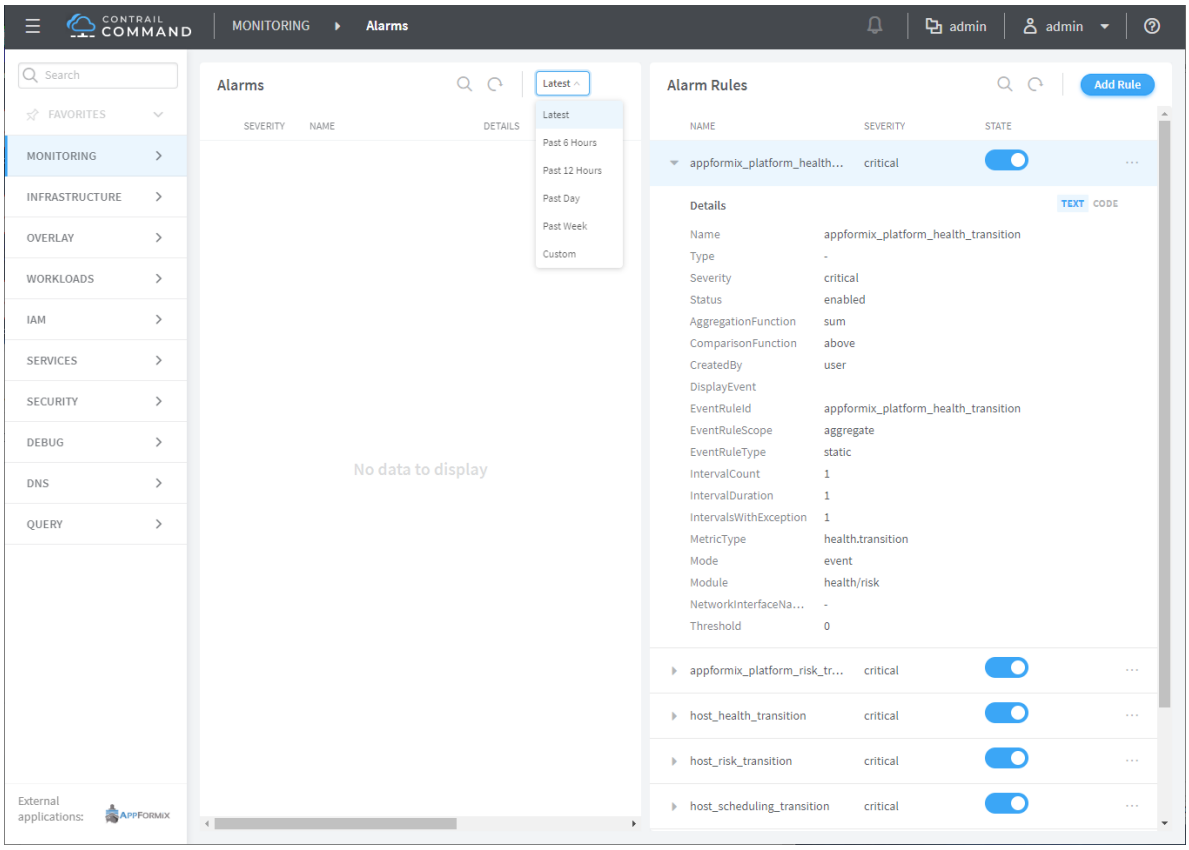


Configuring an Alarm Rule

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.

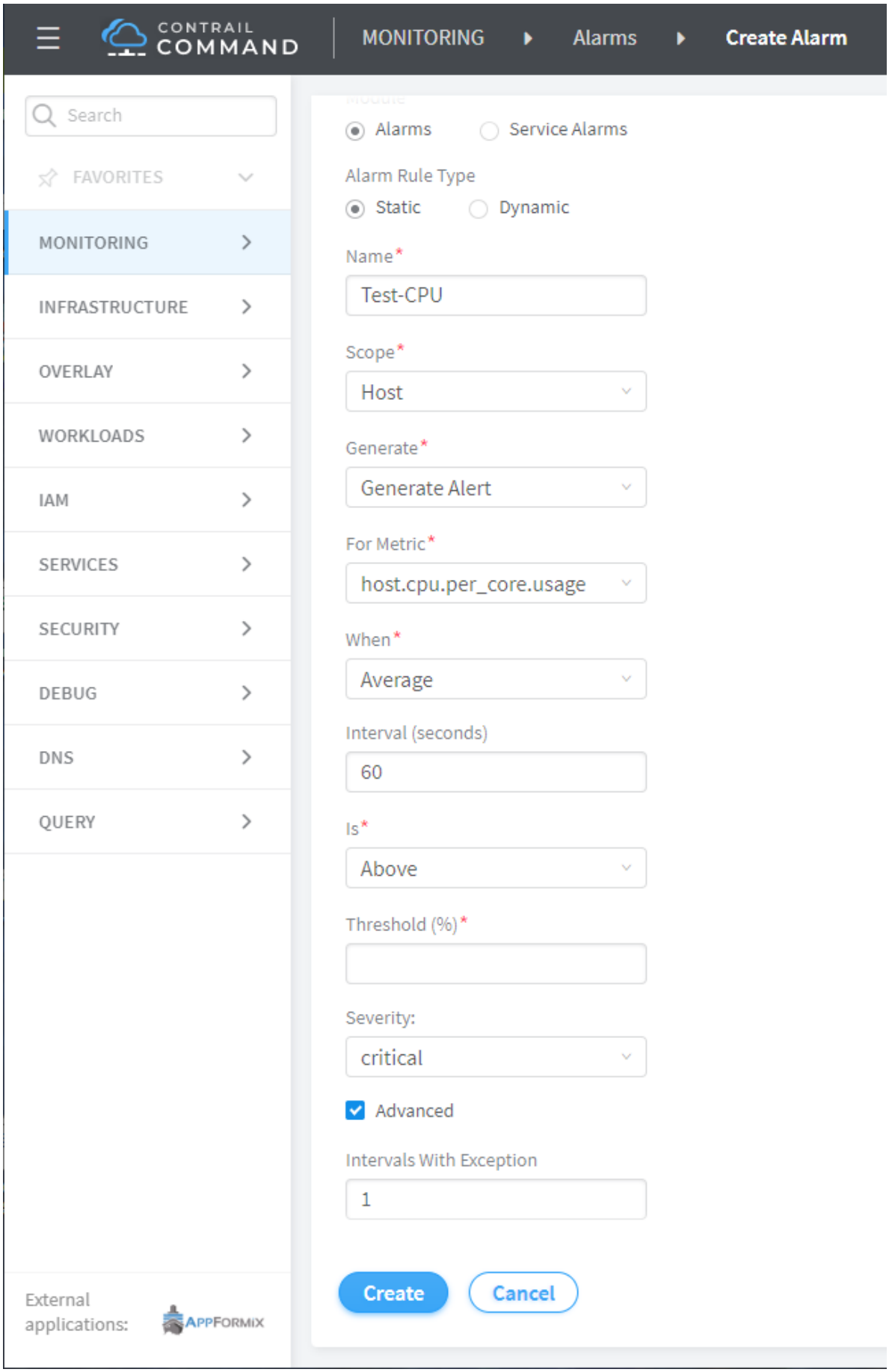
Figure 203: Alarm Active Alerts and Alarm Rules Panel in Contrail Command



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 204: Create and Configure an Alarm in Contrail Command



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 72 on page 388 and descriptions in section "Alarm Definition."

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

RELATED DOCUMENTATION

Configuring Instances in Contrail Insights | 406
Viewing Cluster Node Details and Metric Values | 412
Metrics Collected by Contrail Insights

Configuring Instances in Contrail Insights

IN THIS SECTION

Instance Details Overview | 407
Creating Instances | 407

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

NOTE: For Contrail Insights releases prior to 3.2.6: In order to view and configure instances, your Contrail Insights license subscription must be active.

Instance Details Overview

Table 78 on page 407 provides descriptions for the instances column headers.

Table 78: 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.

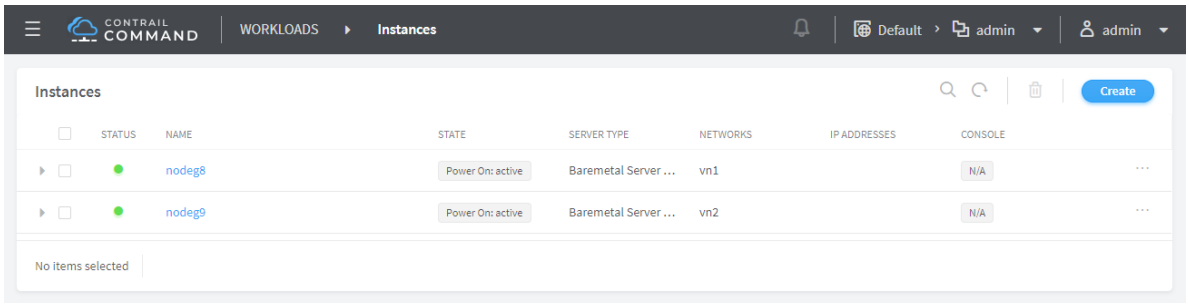
Creating Instances

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 205 on page 408 .

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 205: Workloads > Instances



2. Click **Create**, as shown in Figure 205 on page 408 , 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 79 on page 409 , to define an instance for the selected server:

Figure 206: Create an Instance for a Virtual Machine or New Baremetal Server

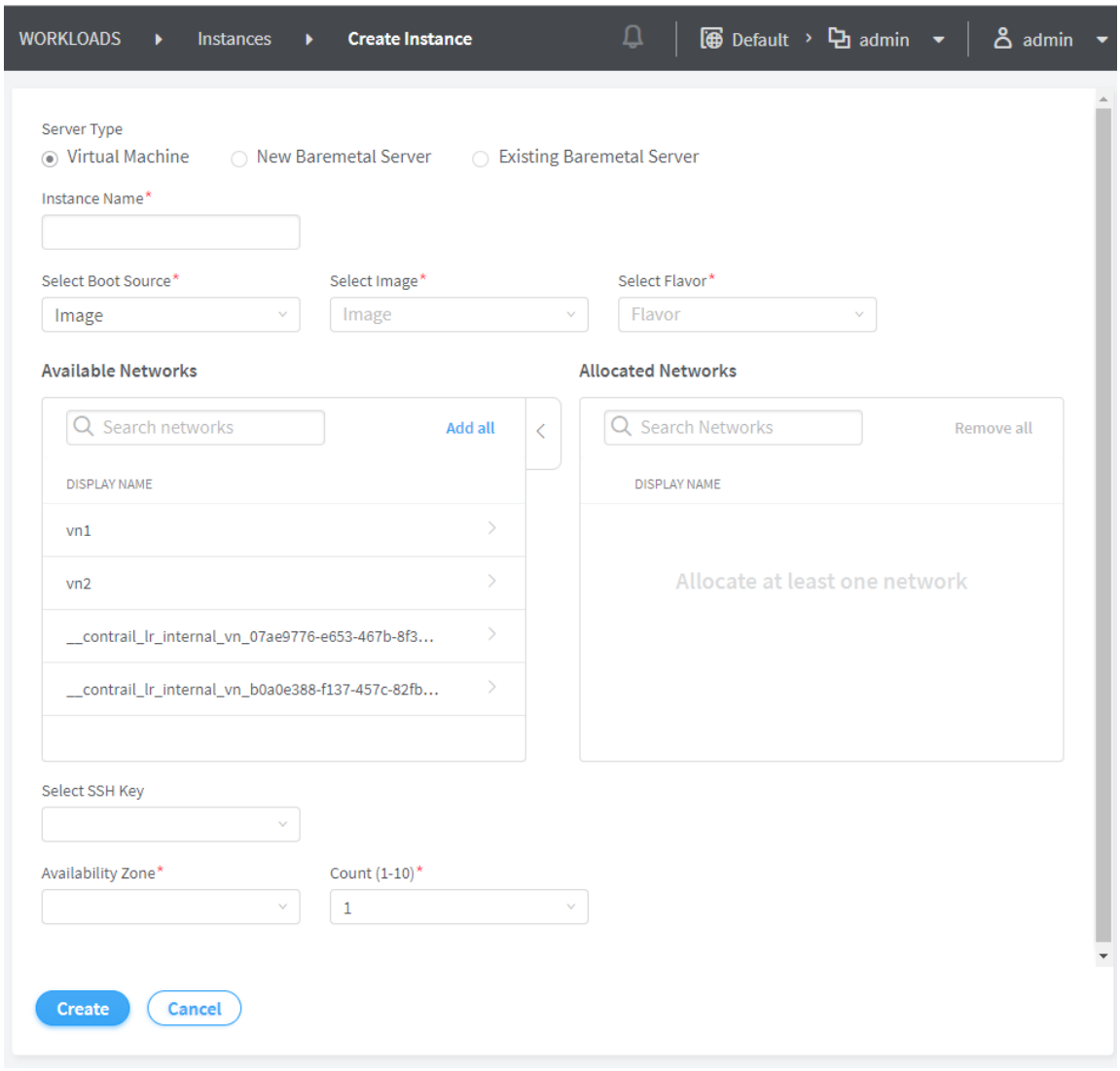


Table 79: Create Instance Fields—Virtual Machine or New Baremetal Server

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.

Table 79: Create Instance Fields—Virtual Machine or New Baremetal Server *(Continued)*

Field	Description
Select Flavor	Select the default configurations for virtual machines.
Available Networks	Network resources that are currently available.
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.
Count	Select a number from 1 - 10, which represents the number of instances to launch.

a. When Existing Baremetal Server is selected, complete the following fields, described in Table 80 on page 411 , to define an instance for the selected server:

Figure 207: Create an Instance for an Existing Baremetal Server

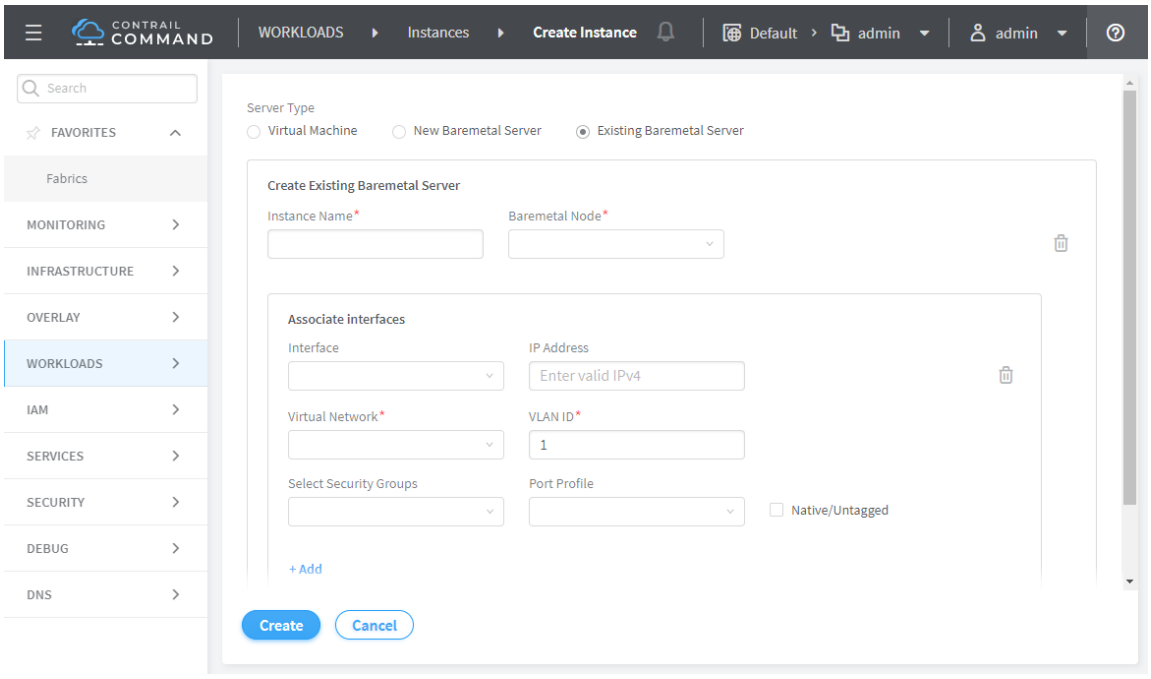


Table 80: Create Instance Fields—Existing Baremetal Server

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.
VLAN ID	Identifier for the VLAN.

Table 80: Create Instance Fields—Existing Baremetal Server *(Continued)*

Field	Description
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 205 on page 408 .

RELATED DOCUMENTATION

Configuring Contrail Insights Alarms using Contrail Command | **381**
Viewing Cluster Node Details and Metric Values | **412**
Metrics Collected by Contrail Insights

Viewing Cluster Node Details and Metric Values

IN THIS SECTION

- Time | **413**
- Legend | **413**
- Chart Data Values | **413**
- Viewing Cluster Node Details and Host Charts | **414**

With cluster node details and host charts, you can view real-time and historical values of all metrics that Contrail Insights 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 Contrail Insights 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 209 on page 415 shows real-time metric values streamed from Contrail Insights.

NOTE: For Contrail Insights releases prior to 3.2.6: In order to view host charts, your Contrail Insights license subscription must be active.

Time

The Time in the Settings dialog box (see Figure 209 on page 415) 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 209 on page 415 . 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 Contrail Insights 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.

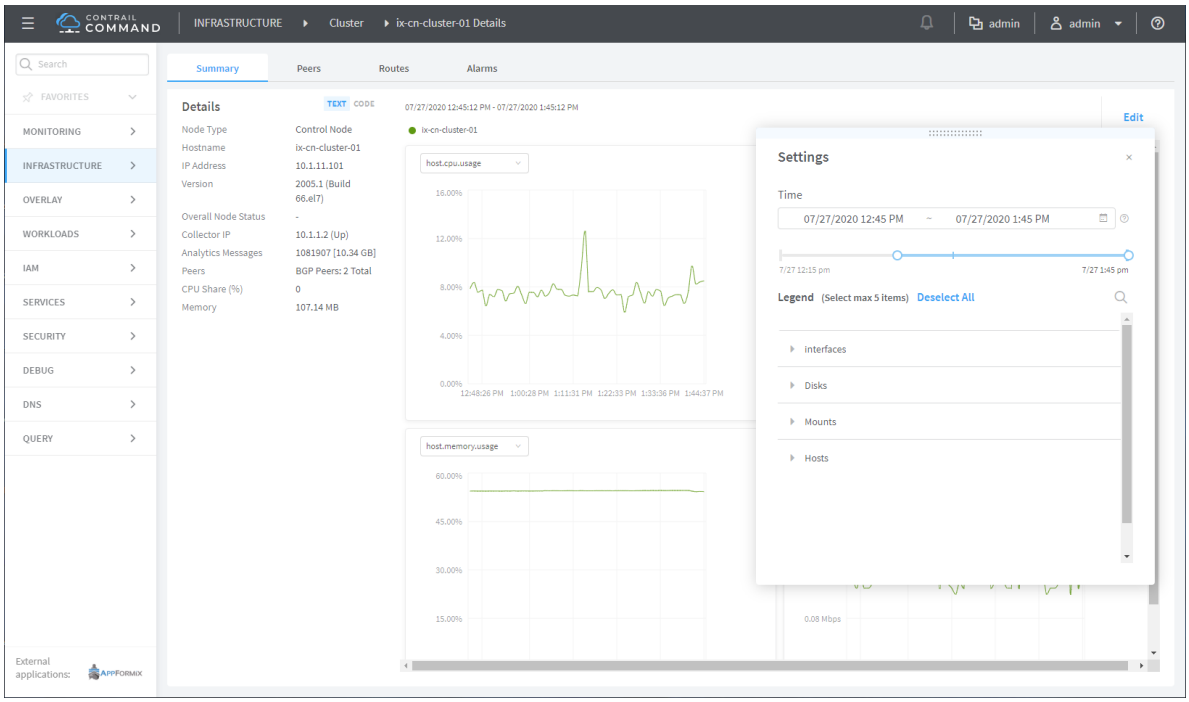
Figure 208: Chart Data Values Tool Tip for a Particular Time

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 209: Real-Time Metric Values Streamed from the Contrail Insights Platform



4. Click **Edit** to launch the Settings dialog box, where you can set the time range and legend. With an active Contrail Insights subscription, you have access to the Contrail Insights 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.

5. Select the Cluster Nodes name for further detail.

   - Peers tab—Includes JSON values, peer type, peer ASN, status, and messages count.

   - Routes tab—Includes routing table JSON values, prefix, protocol, source, next hop, label, security group, and origin virtual-network object (VN).

   - Alarms tab—Includes severity, time, alarm type, and source.

RELATED DOCUMENTATION

- Configuring Contrail Insights Alarms using Contrail Command | **381**
- Contrail Networking Alarms | **117**
- Configuring Instances in Contrail Insights | **406**
- Metrics Collected by Contrail Insights

CHAPTER 7

Common Support Answers

IN THIS CHAPTER

- Debugging Ping Failures for Policy-Connected Networks | 417
- Debugging BGP Peering and Route Exchange in Contrail | 425
- Troubleshooting the Floating IP Address Pool in Contrail | 443
- Removing Stale Virtual Machines and Virtual Machine Interfaces | 472
- Troubleshooting Link-Local Services in Contrail | 476

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 210: Virtual Machine Status Window

Check the tap interface status in the http agent introspect, for example:

```
https://<host ip address>:8085/Snh_ItfReq?name=
```

Figure 211: Tap Interface Status Window

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:

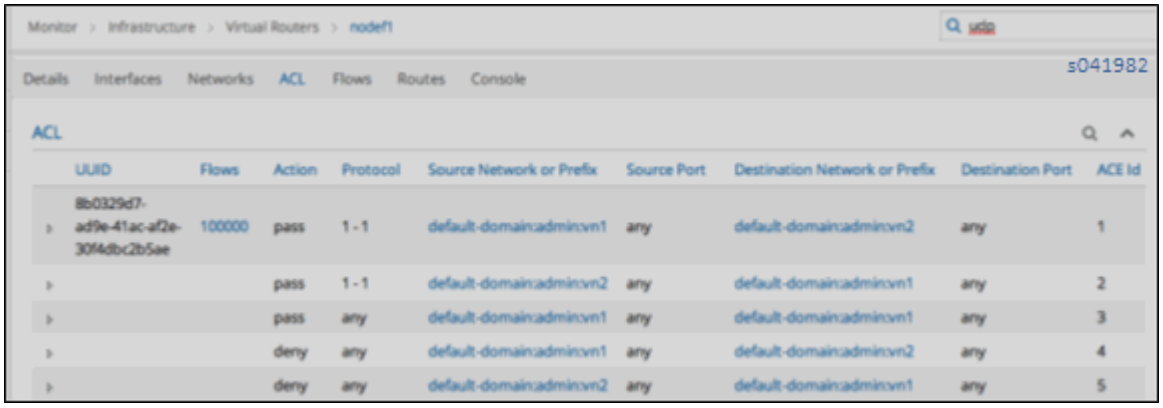
   - Verify that the policy is attached to each of the virtual networks.
   - Each attached policy should have either an explicit rule allowing traffic from one virtual network to the other, or an allow all traffic rule.
   - Verify that the order of the actions in the policy rules is correct, because the 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:

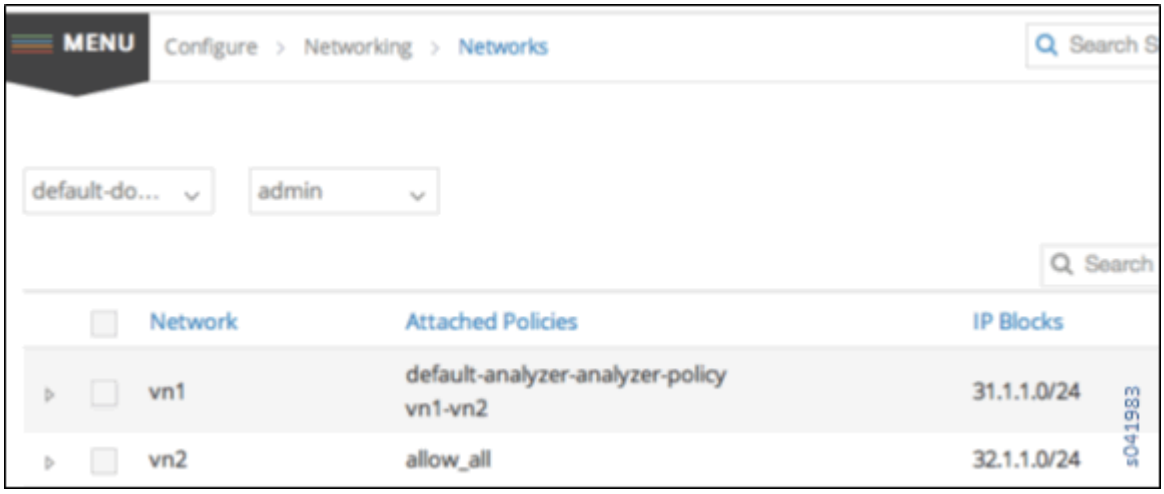
Check VN1-VN2 ACL information from the compute node:

Figure 212: Policies, Attachments, and Traffic Rule Status Window



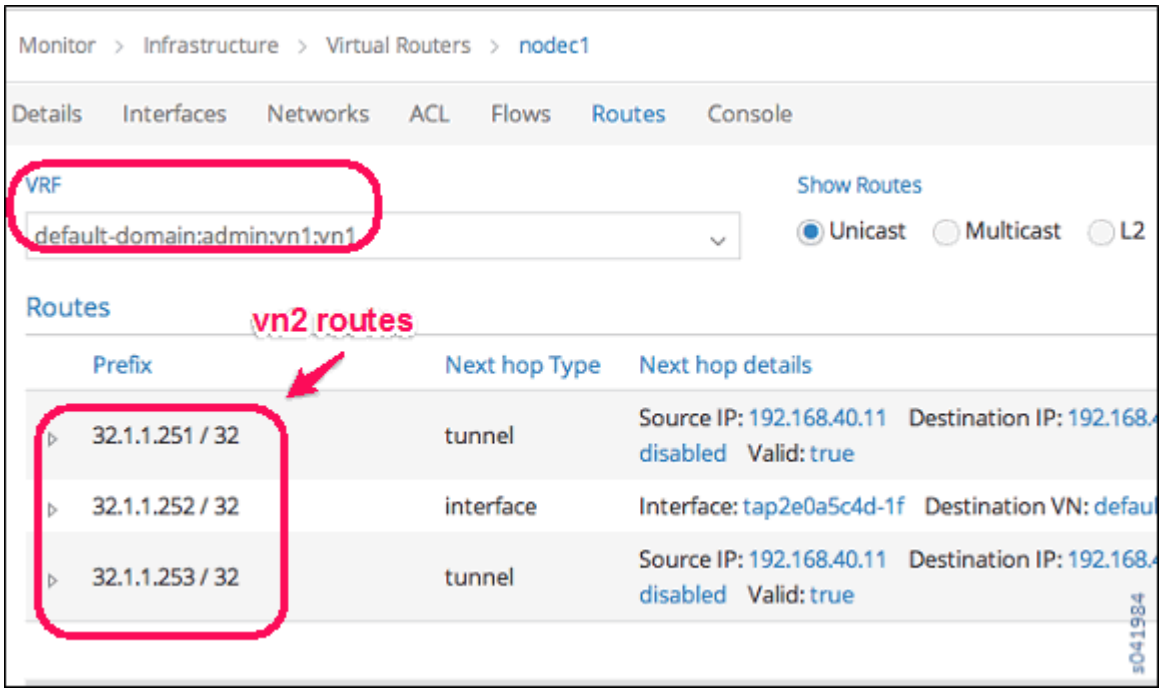
Check the virtual network policy configuration with route information:

Figure 213: Virtual Network Policy Configuration Window



Check the VN1 route information for VN2 routes:

Figure 214: Virtual Network Route Information Window

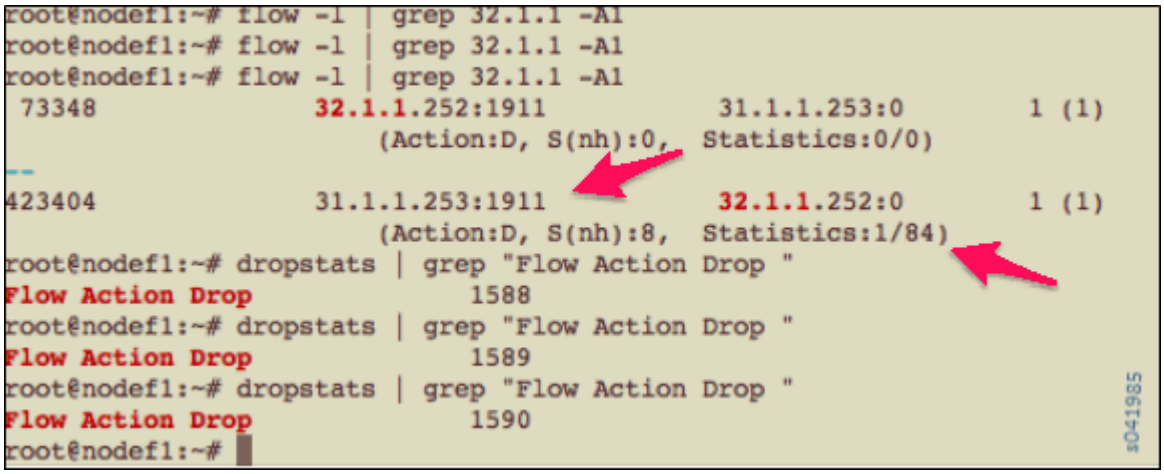


If a route is missing, ping fails. Flow inspection in the compute node displays **Action: D(rop)**.

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 215: Flow and Dropstats Command List



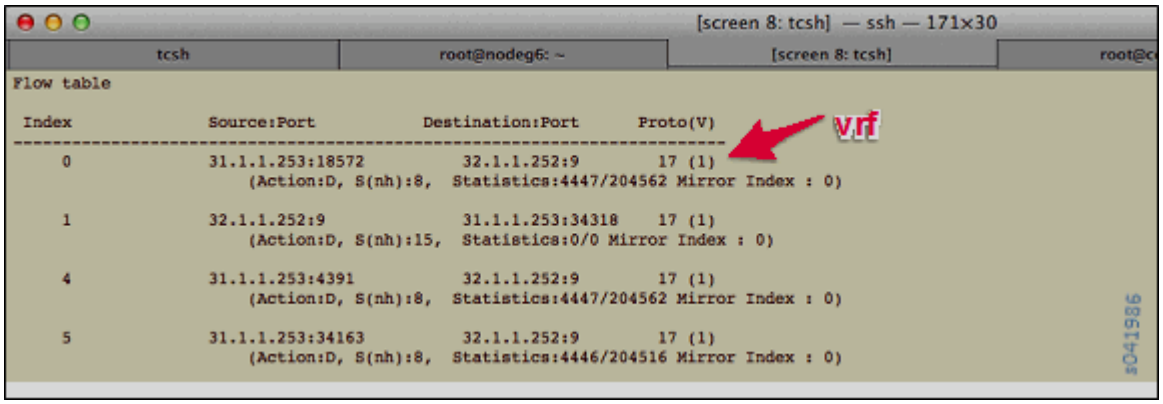
To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

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**

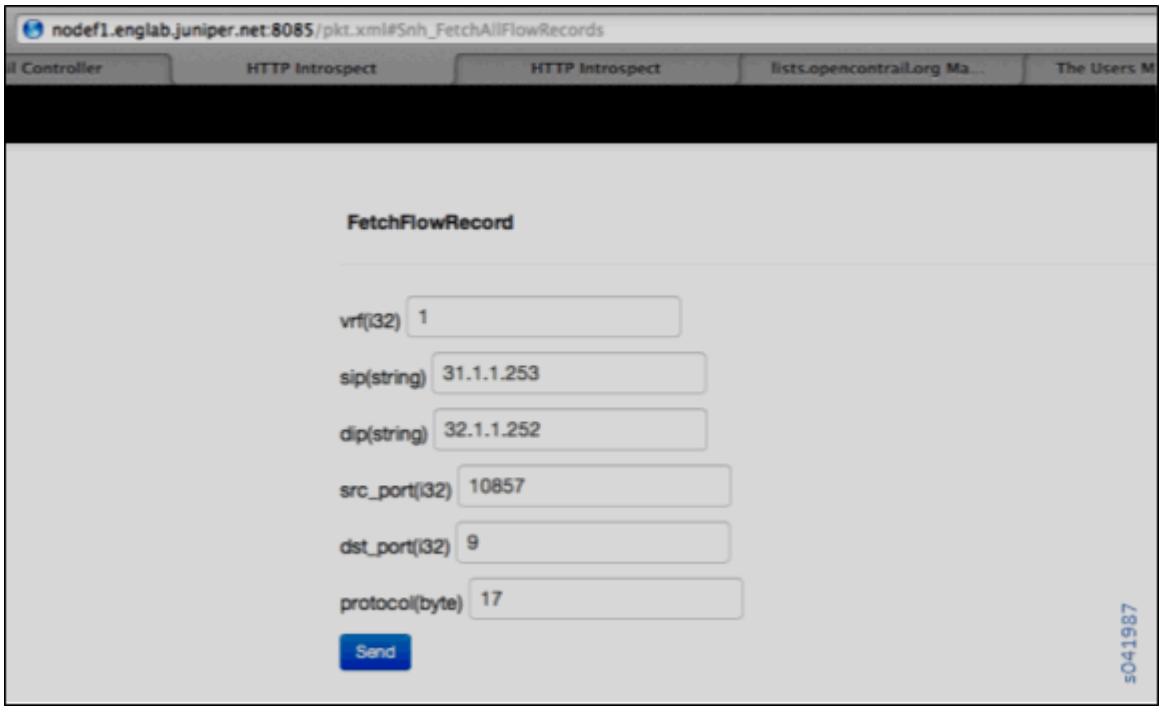
Flow command output:

Figure 216: Flow Command Output Window



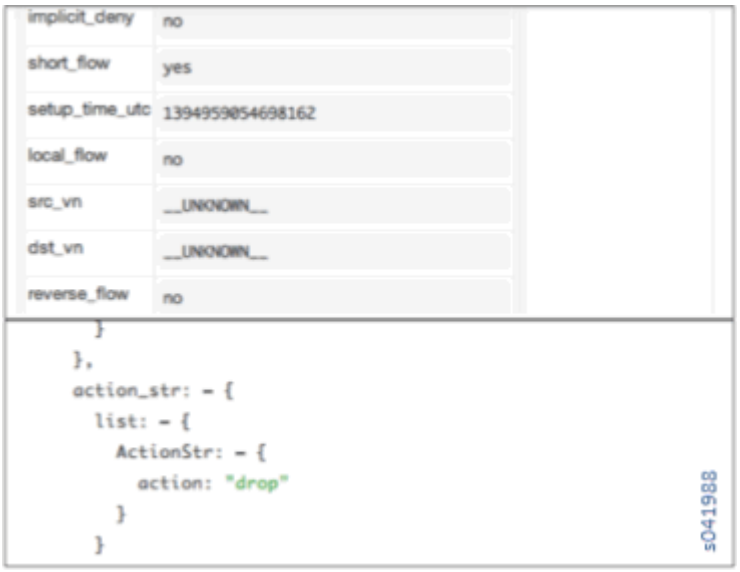
Fetching details of a single flow:

Figure 217: Fetch Flow Record Window



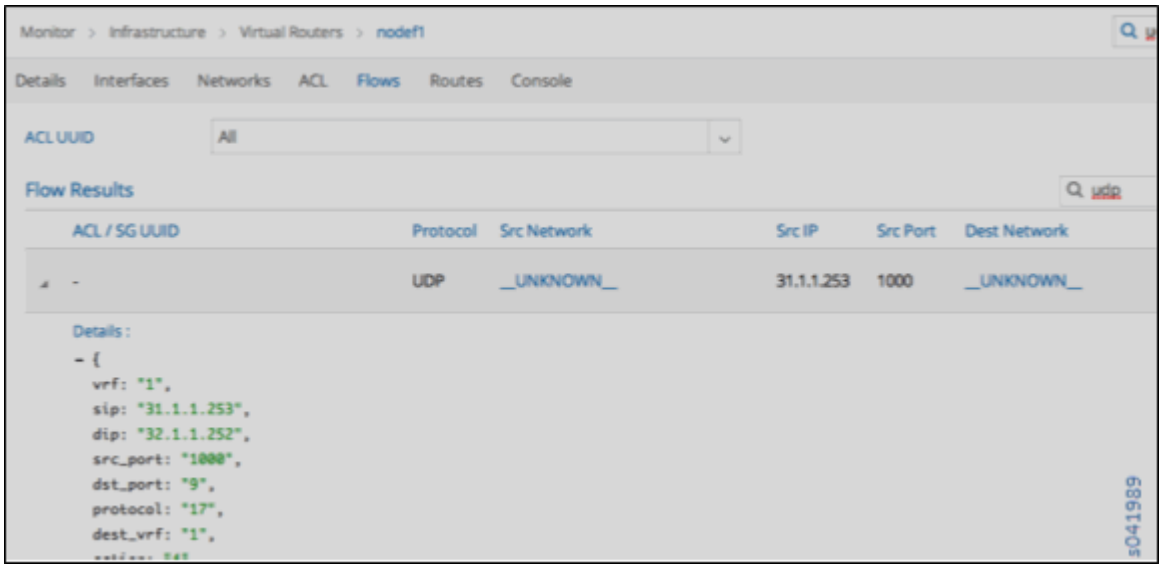
Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 218: Unresolved IP Address Window



You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 219: Unresolved Flow Details Window

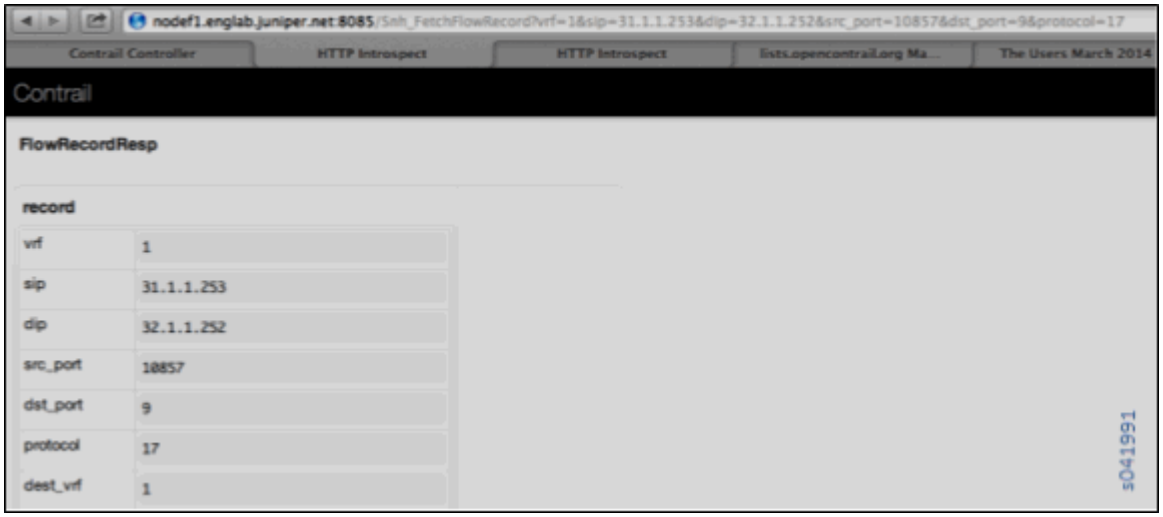


3. Check for protocol-specific network policy action.

   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 220: Protocol-Specific Flow Sample



The following shows that the policy action clearly displays **deny** as the action.

Figure 221: 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 | 425
- Verifying the BGP Routers | 425
- Verifying the Route Exchange | 428
- Debugging Route Exchange with Policies | 431
- Debugging Peering with an MX Series Router | 432
- Debugging a BGP Peer Down Error with Incorrect Family | 434
- Configuring MX Peering (iBGP) | 437
- Checking Route Exchange with an MX Series Peer | 439
- Checking the Route in the MX Series Router | 441

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. List BGP routers with the following Contrail API request.

   `bgp-router` is created in Contrail for each control node, BGPaaS, and external BGP routers. These 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 222: 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"
        }
    ]
}
```


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 223: Sample Output, BGP Router References

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 available on the control node. The control node receives the configuration from Cassandra (starting with Contrail Networking Release 4.0) or from IF-MAP (earlier than Contrail Networking Release 4.0).

   `http: //<host ip address>:8083/Snh_ShowBgpNeighborConfigReq?`

Figure 224: Sample Output, BGP Neighbor Config

```
http: //<host ip address>:8083/Snh_ShowBgpPeeringConfigReq?
```

Figure 225: Sample Output, BGP Peering Config

5. Check the BGP neighbor states on the control node.

```
http: //<host ip address>:8083/Snh_ShowBgpNeighborSummaryReq
```

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 226: Sample Output, Show Routing Instance

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.

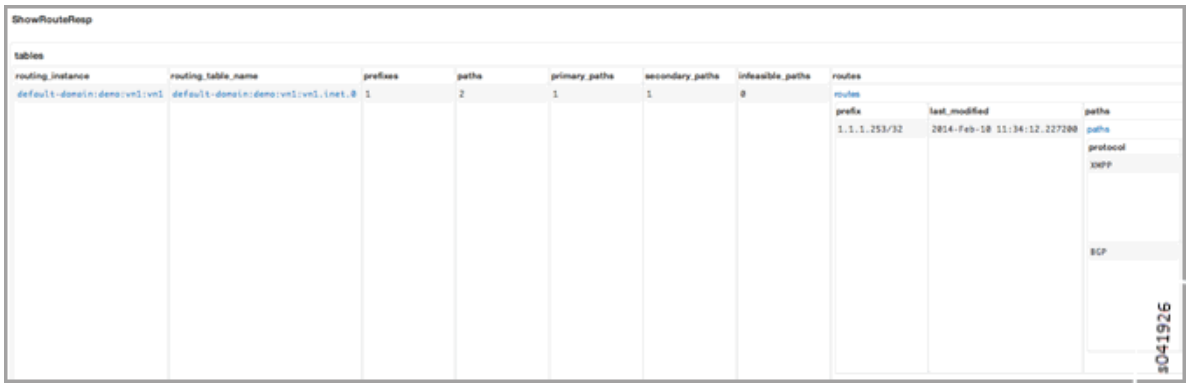
Use the information in Step 2 to validate a route.

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.

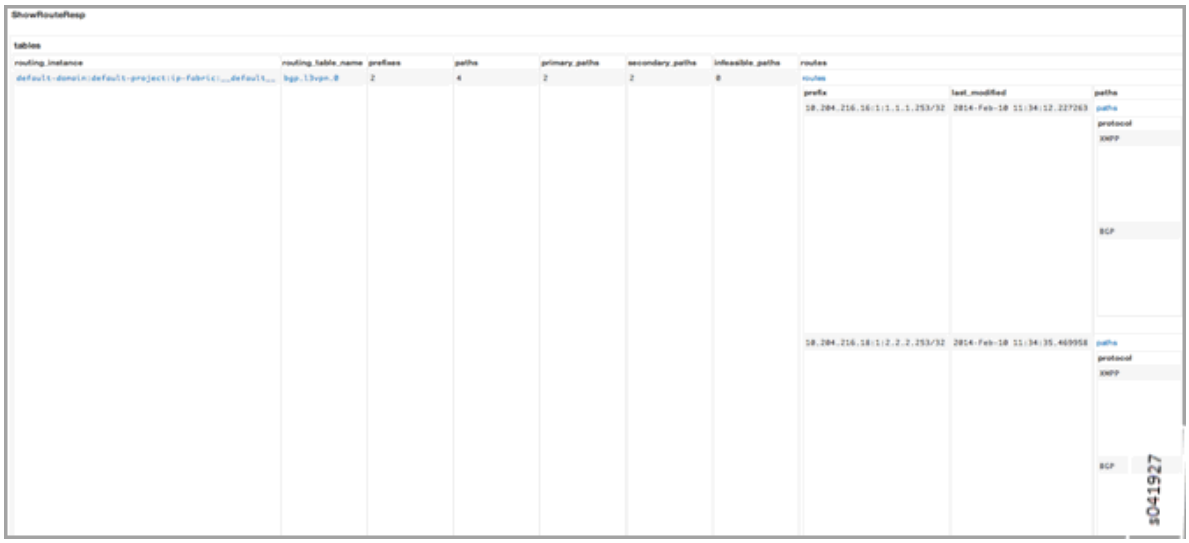
Figure 227: Sample Output, Validate Route



3. Validate the **l3vpn** table.

```
http: //<host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0
```

Figure 228: Sample Output, Validate L3vpn Table

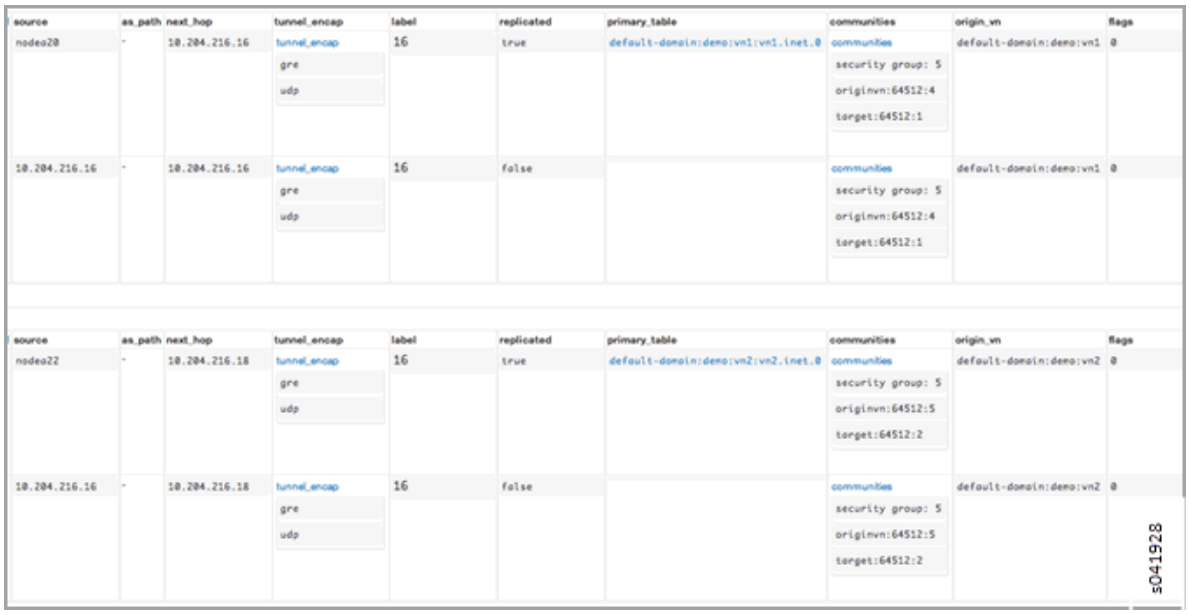


The following sample output has been scrolled horizontally to display the BGP path attributes of each route's. policies.

The extended community (communities column), determines the VRF table to which this VPN route is imported. The **origin_vn** shows the virtual network where this route was created, information useful for applying ACL.

The label (MPLS) and tunnel encap columns can be used for debugging data path issues.

Figure 229: Sample Output, Validate L3vpn Table, Scrolled



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.

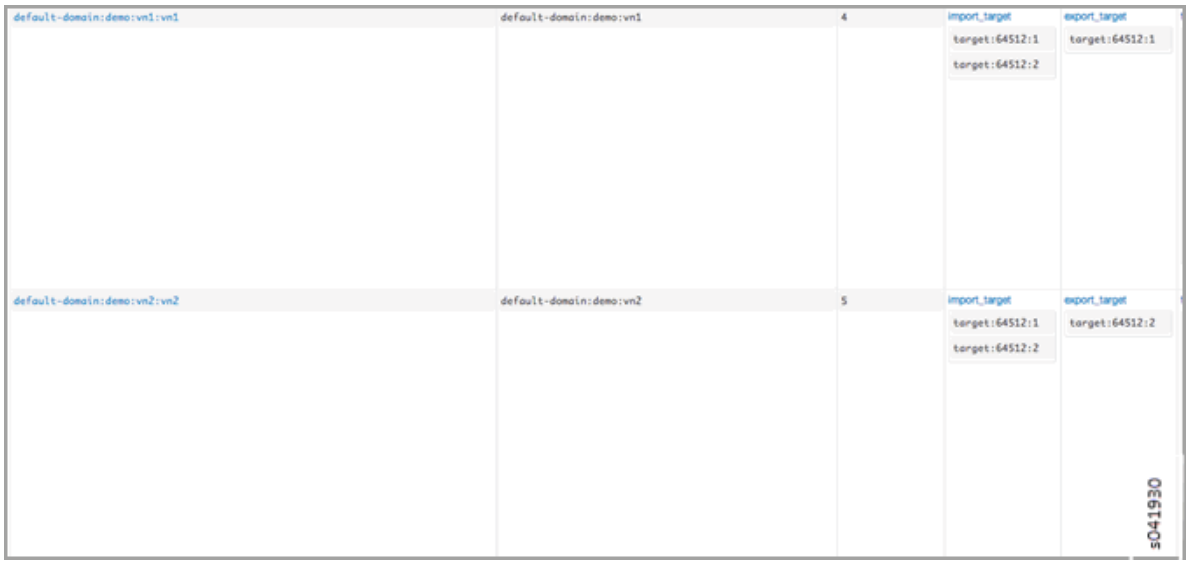
1. Create a network policy to allow vn1 and vn2 traffic and associate the policy to the virtual networks.

Figure 230: Create Policy Window

2. Validate that the routing instances have the correct import_target configuration.

```
http: //<host ip address>:8083/Snh_ShowRoutingInstanceReq?name=
```

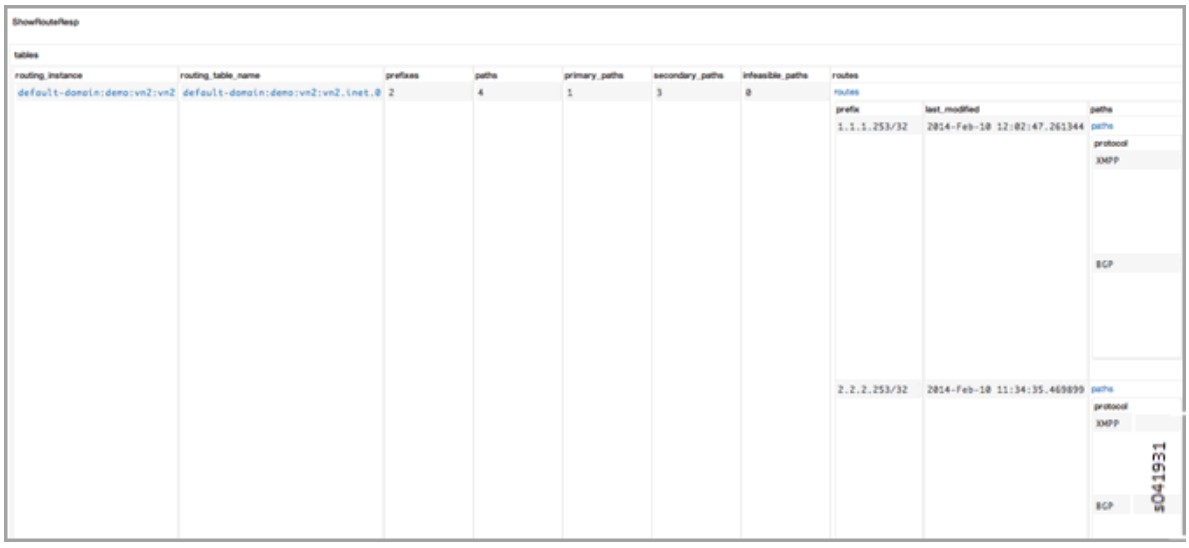
Figure 231: Sample Output, Validate Import Target



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 232: Sample Output, Route Import

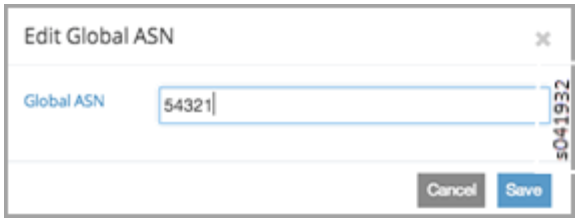


Debugging Peering with an MX Series Router

This section sets up an example BGP MX Series peer and provides some troubleshooting scenarios.

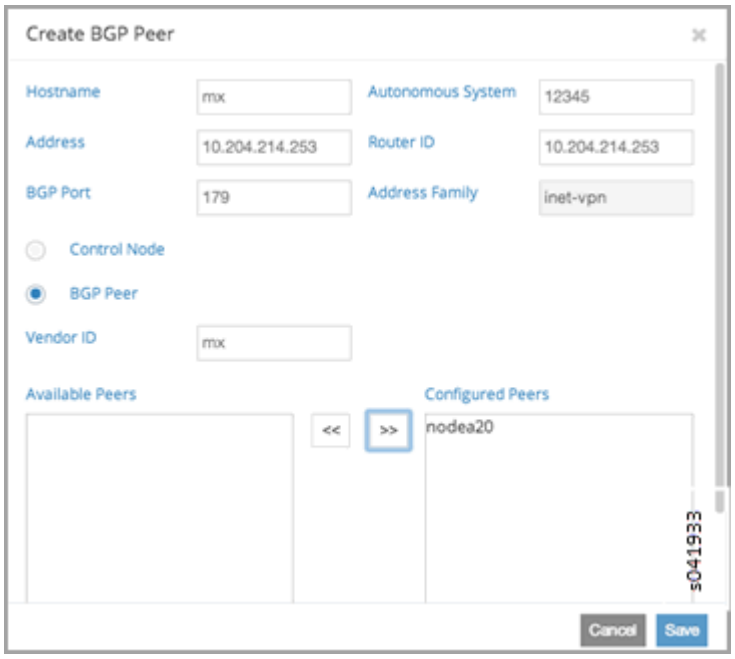
1. Set the Global AS number of the control-node for an MX Series BGP peer, using the Contrail WebUI (eBGP).

Figure 233: Edit Global ASN Window



2. Configure the eBGP peer for the MX Series router. Use the Contrail Web UI or Python provisioning.

Figure 234: 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 235: Sample BGP Peer UVE

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 236: Sample Established BGP Peer UVE

5. Verify the peer status on the MX Series router, using Junos CLI:

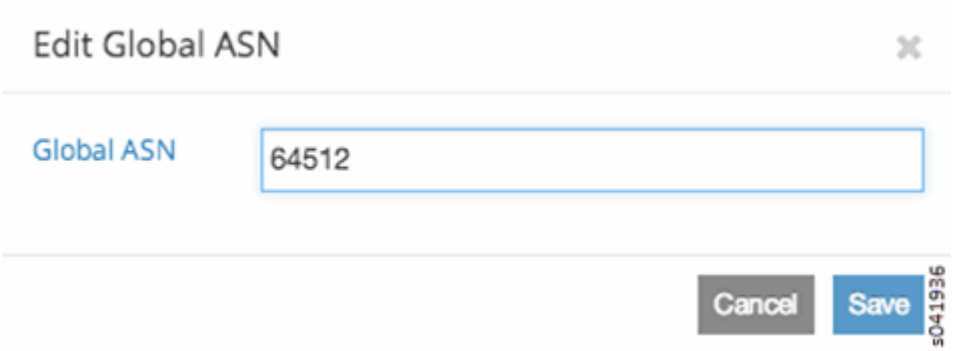
```
run show bgp neighbor <ip address>
Peer: <ip address> AS 54321 Local: <ip address> AS 12345

  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)

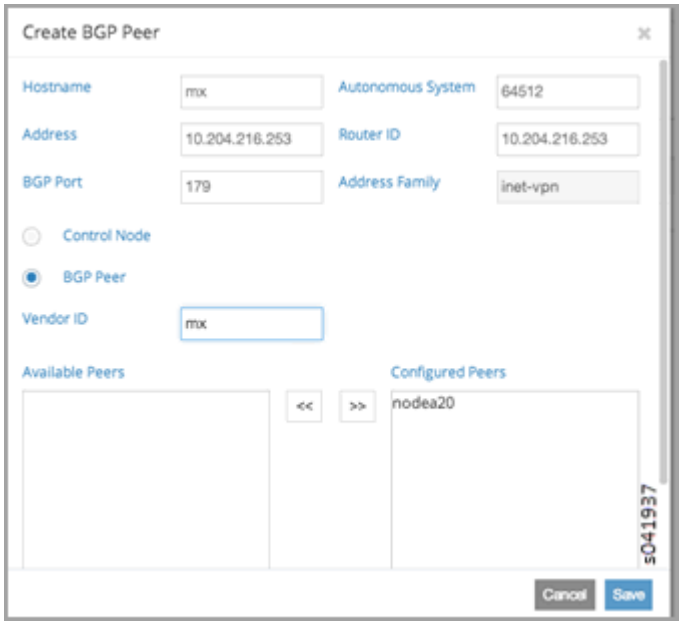
1. Edit the Global ASN.

Figure 237: Edit Global ASN Window



2. Configure the MX Series IBGP peer, using Contrail WebUI or Python provisioning.

Figure 238: 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 239: Sample Established IBGP Peer UVE

4. You can verify the same information at the HTTP introspect page of the control node (8443 in this example).

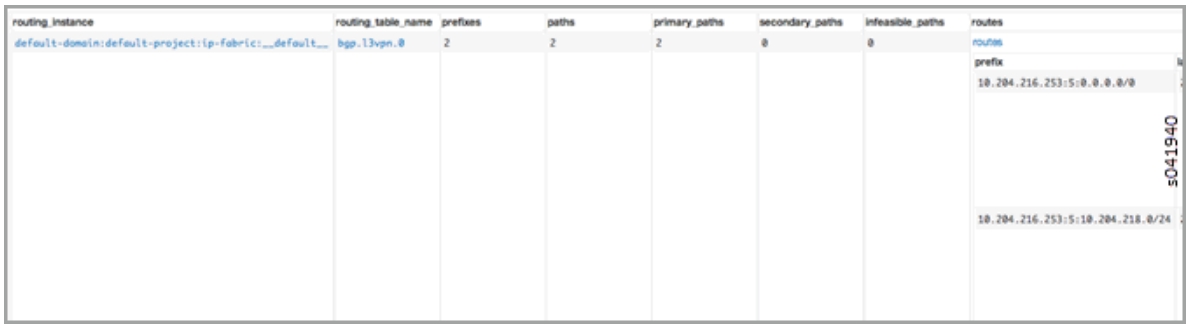
   `http: //<host ip address>:8083/Snh_BgpNeighborReq?ip_address=&domain=`

Figure 240: Sample Established IBGP Peer Introspect Window

Checking Route Exchange with an MX Series Peer

1. Check the route table in the bgp.l3vpn.0 table.

Figure 241: Routing Instance Route Table



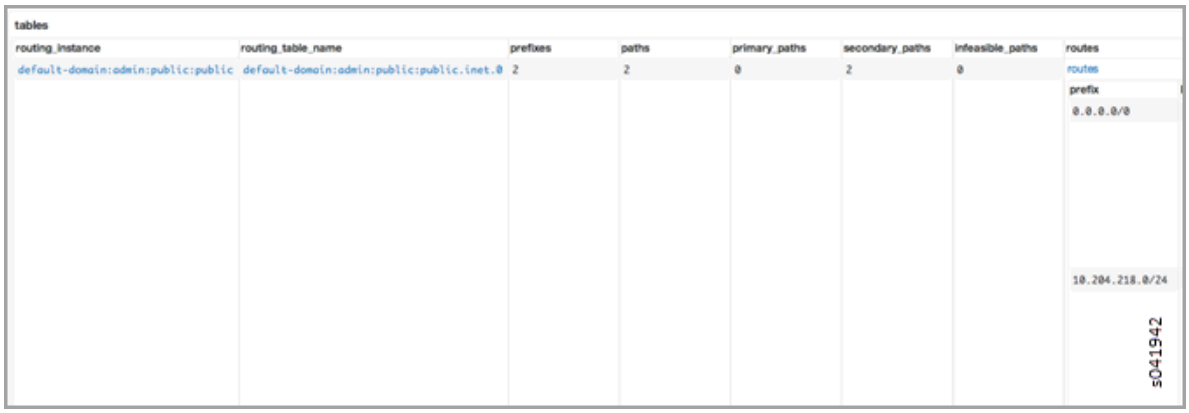
2. Configure a public virtual network.

Figure 242: Routing Instance Route Table

3. Verify the routes in the public.inet.0 table.

```
http: //<host ip address>:8083/Snh_ShowRouteReq?x=default-domain:admin:public:public.inet.0
```

Figure 243: Routing Instance Public IPv4 Route Table



4. Launch a virtual machine in the public network and verify the route in the public.inet.0 table.

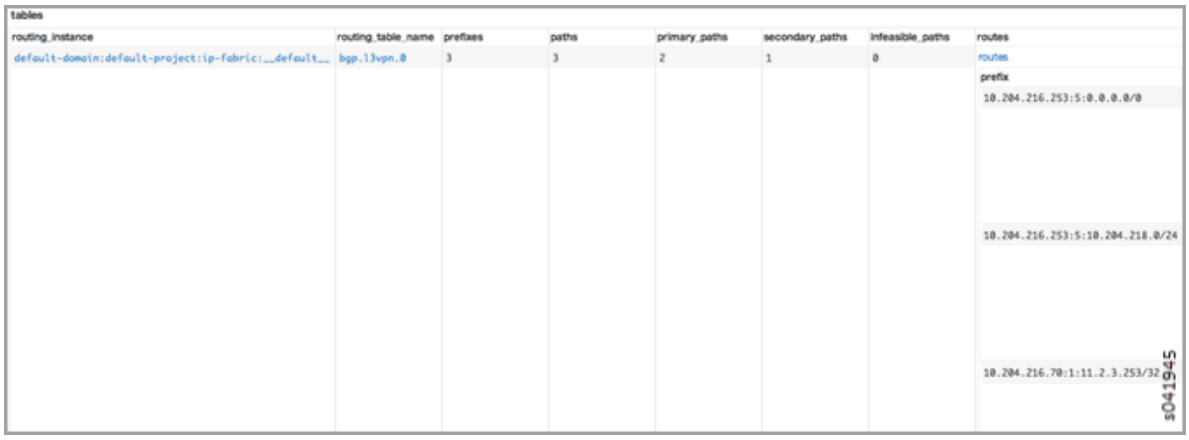
```
http: //<host ip address>:8083/ Snh_ShowRouteReq?x=default-domain:admin:public:public.inet.0
```

Figure 244: Virtual Machine Routing Instance Public IPv4 Route Table

5. Verify the route in the bgp.l3vpn.0 table.

```
http: //<host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0
```

Figure 245: BGP Routing Instance Route Table



Checking the Route in the MX Series Router

Use Junos CLI show commands from the router to check the route. These commands assume that the routing instance with the imported route table from Contrail is configured on the MX Series router, either manually or by using Device Manager.

```
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 | **444**
- Example | **445**
- Example: MX80 Configuration for the Gateway | **446**
- Ping the Floating IP from the Public Network | **449**
- Troubleshooting Details | **450**
- Get the UUID of the Virtual Network | **450**
- View the Floating IP Object in the API Server | **451**
- View floating-ips in floating-ip-pools in the API Server | **455**
- Check Floating IP Objects in the Virtual Machine Interface | **458**
- View the BGP Peer Status on the Control Node | **462**
- Querying Routes in the Public Virtual Network | **463**
- Verification from the MX80 Gateway | **465**
- Viewing the Compute Node Vnsw Agent | **467**

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:

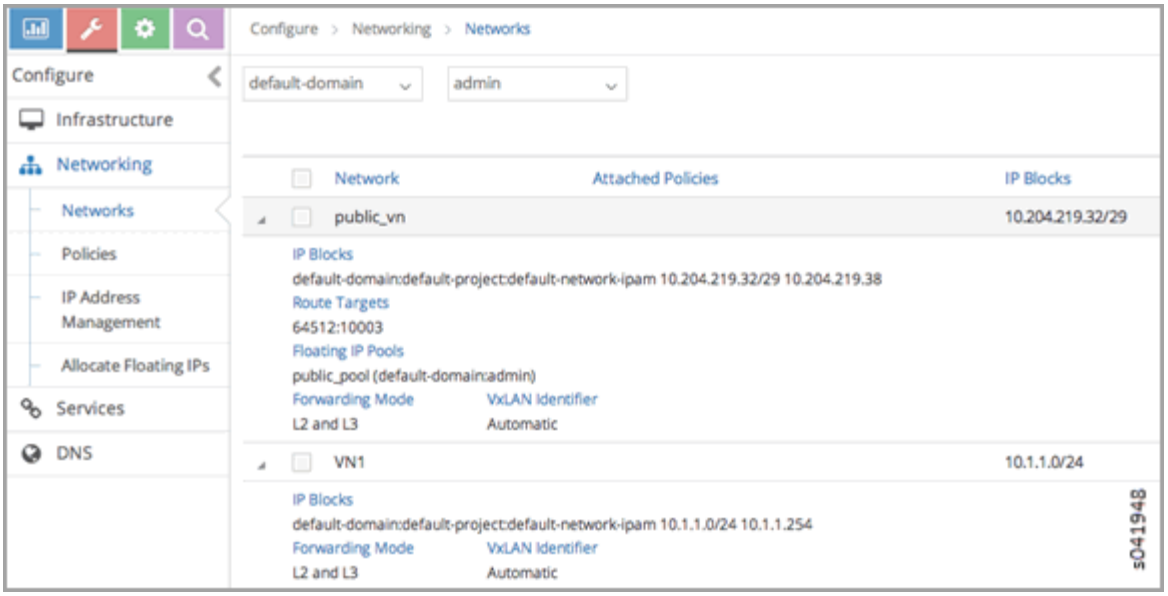
Public virtual network:

- 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`

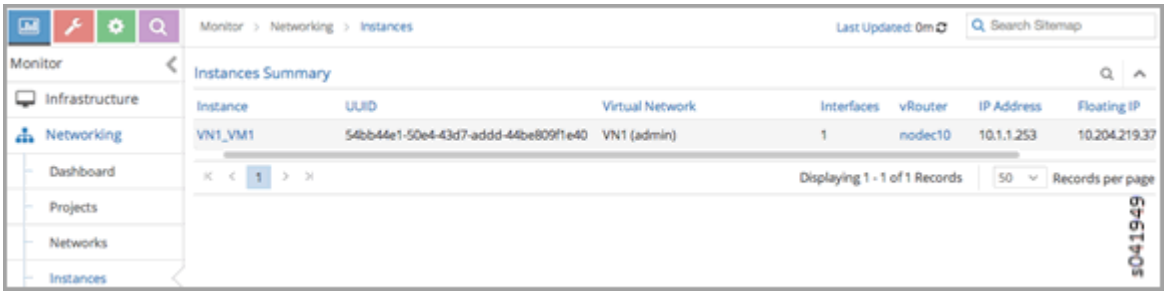
Private virtual network:

- Virtual network name: `vn1`
- Subnet: `10.1.1.0/24`

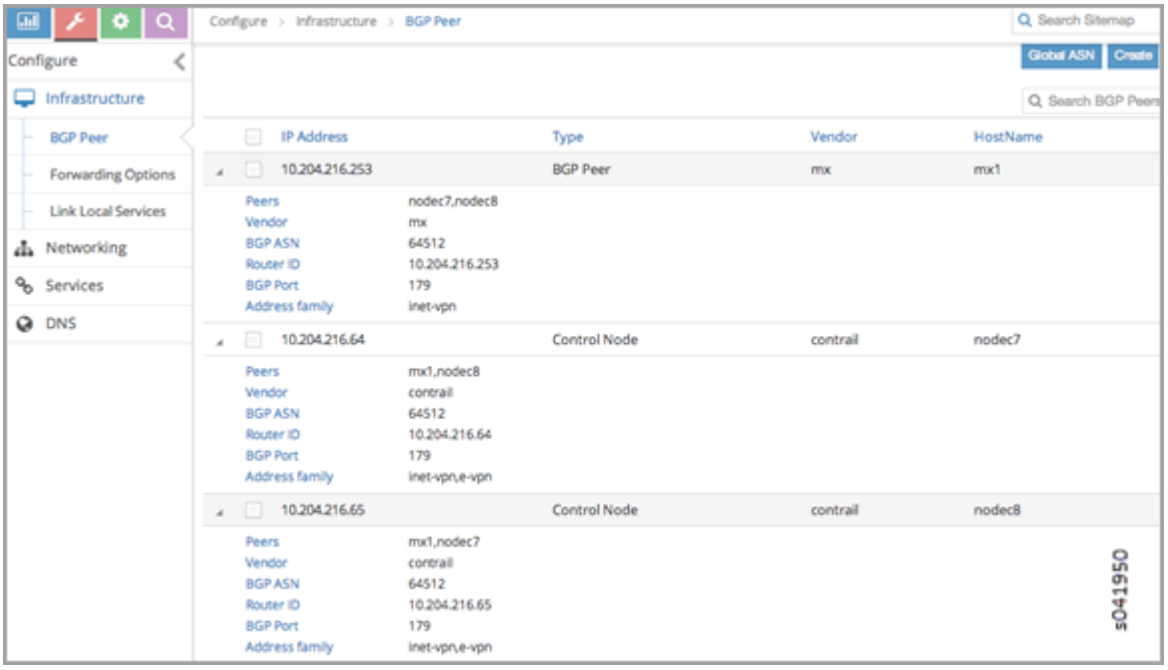
Example



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 Contrail Networking.

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; openstack network list) 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 -s -X GET -H "X-Auth-Token: $(openstack token issue | grep '| id' | awk '{print $4}')" 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",
            "description": null,
            "enable": true,
            "last_modified": "2014-02-07T10:06:42.234423",
            "permissions": {
                "group": "admin",
                "group_access": 7,
                "other_access": 7,
                "owner": "admin",
                "owner_access": 7
            },
            "uuid": {
                "uuid_lslong": 9284482284331406877,
```

```
                "uuid_mslong": 4859515279882014024

            }

        },

        "name": "public_vn",

        "network_ipam_refs": [

            {

                "attr": {

                    "ipam_subnets": [

                        {

                            "default_gateway": "10.204.219.38",

                            "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 `openstack portlist` to get the virtual machine interface:

  Example

```
[root@nodec6 ~]# openstack portlist

+--------------------------------------
+--------------------------------------------------------------------------------+
```

```
| 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"

            }

        ]

    }

}
```

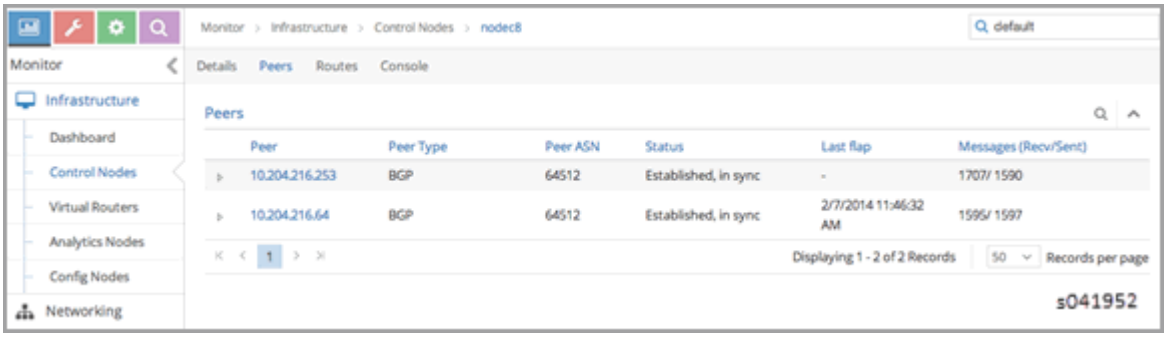
View the BGP Peer Status on the Control Node

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:



- Using the control-node Introspect:

  ```
  http://nodec7:8083/Snh_BgpNeighborReq?ip_address=&domain=
  ```

  ```
  http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=
  ```

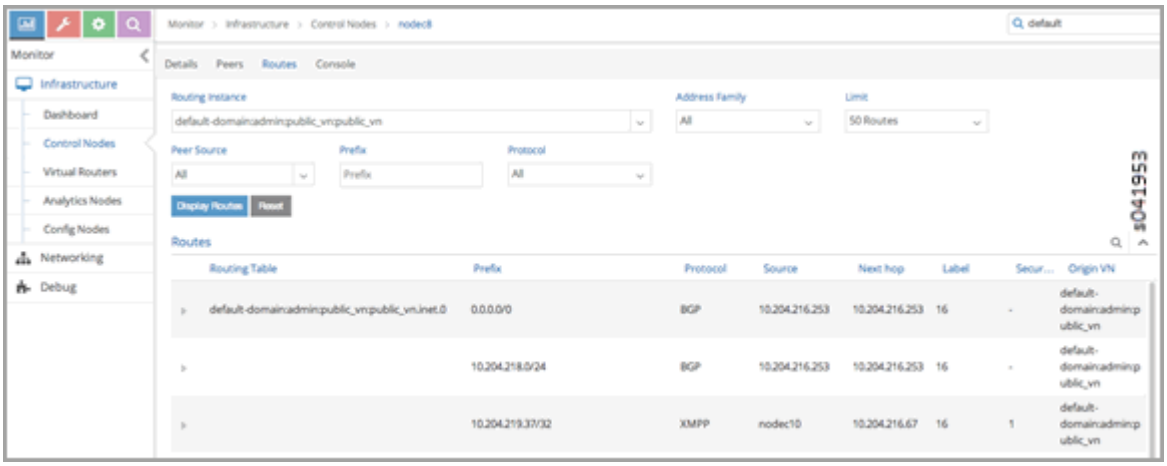
Querying Routes in the Public Virtual Network

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:



- 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

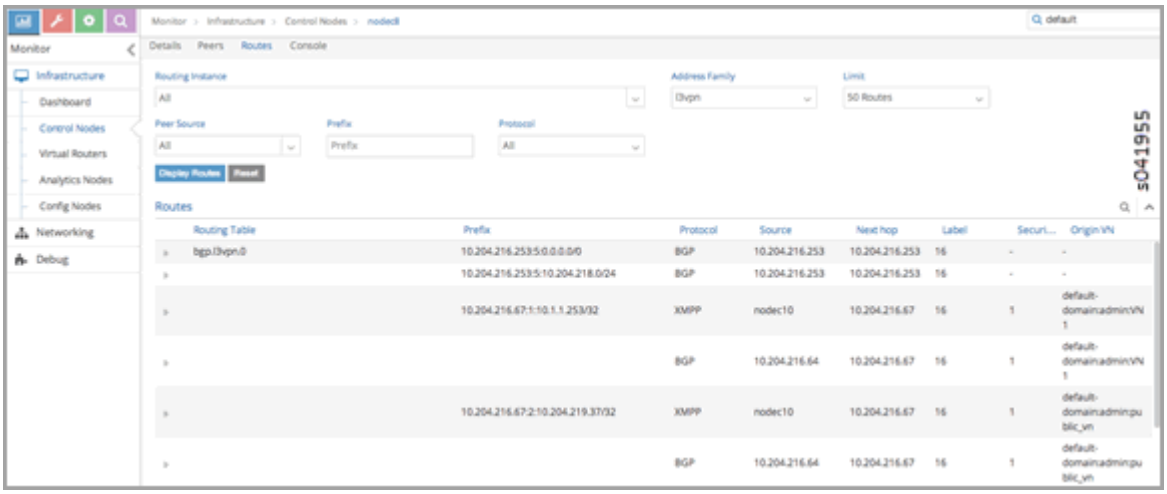
```
http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=
```

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.

Example

- Using the Contrail UI:



- 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

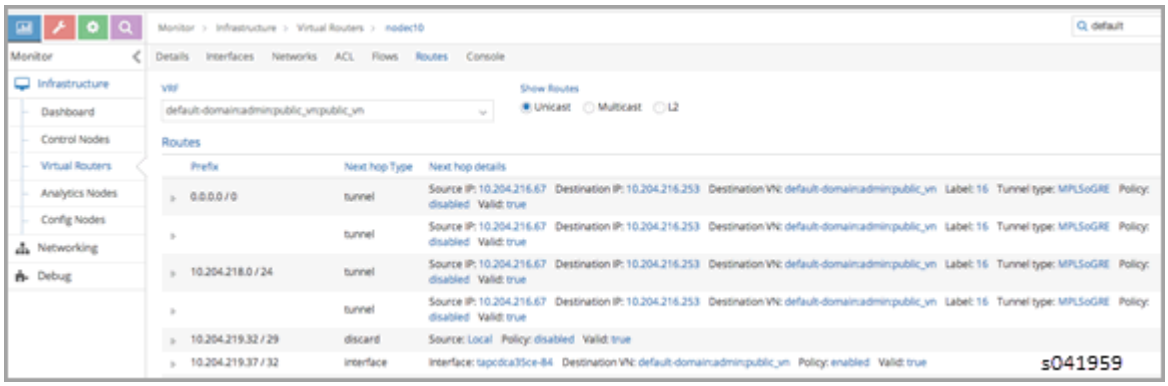
The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

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

1. 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

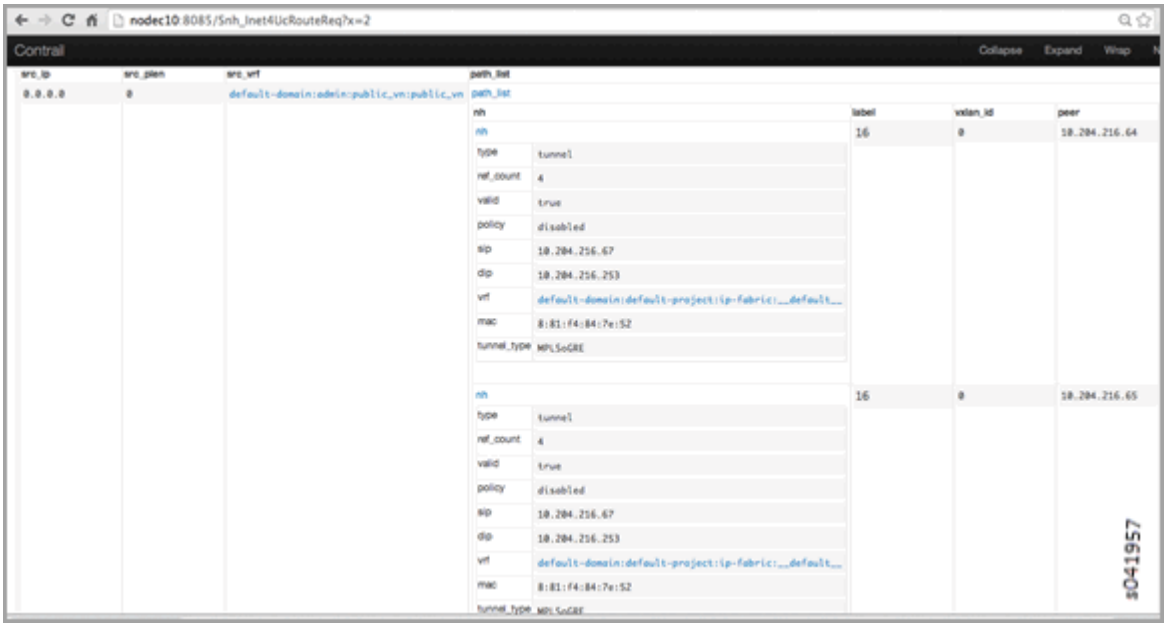
Example

1. In the following example, the unicast route table index is 2.

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



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_KDropStatsReq?`

  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 UDP and 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 p1p0p0 host 10.204.216.253 -vv
tcpdump: WARNING: p1p0p0: no IPv4 address assigned
tcpdump: listening on p1p0p0, 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 | **472**
- Show Virtual Machines | **474**
- Delete Methods | **475**

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.

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 | 476
- Troubleshooting Procedure for Link-Local Services | 476
- Metadata Service | 478
- Troubleshooting Procedure for Link-Local Metadata Service | 478

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**.

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:

   Checking the tap interface status in the http agent introspect:

   `http://<compute-node-ip>:8085/Snh_ItfReq?name=`

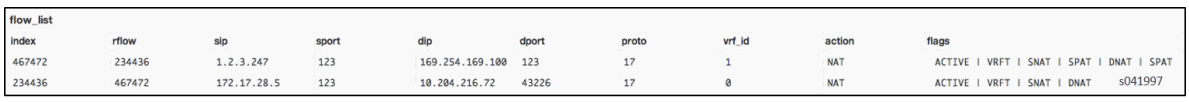
3. Check the link-local configuration in the vrouter agent. Make sure the configured link-local service is displayed.

   `http://<compute-node-ip>:8085/Snh_LinkLocalServiceInfo?`

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.



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.

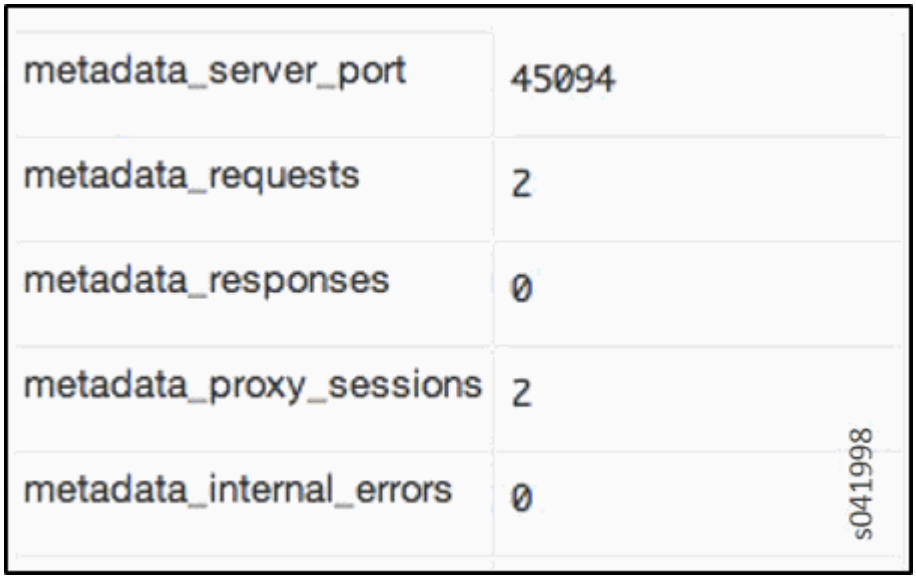
However, for metadata service, the flow is always set up to the compute node, so the vrouter agent will update and proxy the HTTP request. The vrouter agent listens on a local port to receive the metadata requests. Consequently, the reverse flow has the compute node as the source IP, the local port on 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.



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`

2

PART

Contrail Commands and APIs

Contrail Commands | 481

Contrail Application Programming Interfaces (APIs) | 501

CHAPTER 8

Contrail Commands

IN THIS CHAPTER

- Getting Contrail Node Status | 481
- contrail-logs (Accessing Log File Messages) | 493
- contrail-status (Viewing Node Status) | 496
- contrail-version (Viewing Version Information) | 498

Getting Contrail Node Status

IN THIS SECTION

- Overview | 481
- UVE for NodeStatus | 482
- Node Status Features | 482
- Using Introspect to Get Process Status | 489
- contrail-status script | 491

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" aggtype="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 | **493**
- Option Descriptions | **494**
- Example Uses | **495**

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 {ObjectVNTable, ObjectVMTable, ObjectSITable, ObjectVRouter,
ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection,
ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode}]
                    [--object-id OBJECT_ID]
                    [--object-select-field {ObjectLog,SystemLog}]
                    [--trace TRACE]
```

Option Descriptions

The following are the descriptions for each of the option arguments available for `contrail-logs`.

```
optional arguments:
  -h, --help
                        show this help message and exit
  --opserver-ip OPSERVER_IP
                        IP address of OpServer (default: 127.0.0.1)
  --opserver-port OPSERVER_PORT
                        Port of OpServer (default: 8081)
  --start-time START_TIME
                        Logs start time (format now-10m, now-1h) (default: now-10m)
  --end-time END_TIME
                        Logs end time (default: now)
  --last LAST
                        Logs from last time period (format 10m, 1d) (default: None)
  --source SOURCE
                        Logs from source address (default: None)
  --module {ControlNode, VRouterAgent, ApiServer, Schema, OpServer, Collector, QueryEngine,
ServiceMonitor, DnsAgent}
```

```
                        Logs from module (default: None)
  --category CATEGORY
                        Logs of category (default: None)
  --level LEVEL
                        Logs of level (default: None)
  --message-type MESSAGE_TYPE
                        Logs of message type (default: None)
  --reverse
                        Show logs in reverse chronological order (default: False)
  --verbose
                        Show internal information (default: True)
  --all
                        Show all logs (default: False)
  --object {ObjectVNTable, ObjectVMTable, ObjectSITable, ObjectVRouter, ObjectBgpPeer,
ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection, ObjectCollectorInfo,
ObjectGeneratorInfo, ObjectConfigNode}
                        Logs of object type (default: None)
  --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 messagess 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 ObjectVNTable --object-id demo:admin:vn1
   ```

   `--object` accepts the following values - `ObjectVNTable, 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 "2020 May 12 18:30:27.0" --end-time "2020 May 12 18:31:27.0"
   ```

contrail-status (Viewing Node Status)

IN THIS SECTION

- Syntax | **497**
- Description | **497**
- Required Privilege Level | **497**
- Sample Output | **497**
- Release Information | **498**

Syntax

```
[root@host ~]# contrail-status
```

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
```

Release Information

Command introduced in Contrail Release 1.0.

contrail-version (Viewing Version Information)

IN THIS SECTION

- Syntax | **499**
- Description | **499**
- Required Privilege Level | **499**
- Sample Output | **499**
- Sample Output | **500**
- Release Information | **500**

Syntax

```
[root@host]# contrail-version
```

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                                 Version                 Build-ID | Repo | RPM Name
--------------------------------------- ----------------------- ---------------------------------
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                                 Version                 Build-ID | Repo | RPM Name
--------------------------------------- ----------------------- ---------------------------------
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
```

Release Information

Command introduced in Contrail Release 1.0.

CHAPTER 9

Contrail Application Programming Interfaces (APIs)

IN THIS CHAPTER

- Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs) | 501
- Log and Flow Information APIs | 515
- Working with Neutron | 523
- Support for Amazon VPC APIs on Contrail OpenStack | 527

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

IN THIS SECTION

- User-Visible Entities | 502
- Common UVEs in Contrail | 503
- Virtual Network UVE | 503
- Virtual Machine UVE | 504
- vRouter UVE | 504
- UVEs for Contrail Nodes | 505
- Wild Card Query of UVEs | 505
- Filtering UVE Information | 505

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 the Contrail Networking API Reference Guide.

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:

1. `sfilt` : filter by source (usually the hostname of the generator)
2. `mfilt` : filter by module (the module name of the generator)
3. `cfilt` : filter by content, useful when only part of a UVE needs to be retrieved
4. `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"
                                }
                            }
                        ]
                    }
```

```
                },
                "a3s19:VRouterAgent"
            ]
        ]
    }
}
```

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 | 515

Log and Flow Information APIs

IN THIS SECTION

- HTTP GET APIs | 515
- HTTP POST API | 516
- POST Data Format Example | 516
- Query Types | 518
- Examining Asynchronous Query Status | 518
- Examining Query Chunks | 519
- Example Queries for Log and Flow Data | 519

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:

1. SELECT *field1, field2* ...
2. FROM *table1*
3. WHERE *field1* = *value1* AND *field2* = *value2* ...
4. FILTER BY ...
5. SORT BY ...
6. 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:

1. `start_time` — the start of the time period
2. `end_time` — the end of the time period
3. `table` — the table from which to extract data
4. `select_fields` — the columns to display in the extracted data
5. `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, ObjectVNTable,
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. The result is "202 Accepted." This status code indicates the request has been accepted for processing but the processing has not been completed.

Examining Asynchronous 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 | **523**
- Network Sharing in Neutron | **524**
- Commands for Neutron Network Sharing | **525**
- Support for Neutron APIs | **525**
- Contrail Neutron Plugin | **526**
- DHCP Options | **526**
- Incompatibilities | **527**

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.

Table 81: Neutron commands

Action	Command
List all shared networks.	`neutron net-list --shared`
Create a network that has the shared attribute.	`neutron net-create <net-name> -shared`
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 | **528**
- Mapping Amazon VPC Features to OpenStack Contrail Features | **528**
- VPC and Subnets Example | **529**
- Euca2ools CLI for VPC and Subnets | **530**
- Security in VPC: Network ACLs Example | **530**
- Euca2ools CLI for Network ACLs | **532**
- Security in VPC: Security Groups Example | **532**
- Euca2ools CLI for Security Groups | **533**
- Elastic IPs in VPC | **534**
- Euca2ools CLI for Elastic IPs | **534**
- Euca2ools CLI for Route Tables | **535**
- Supported Next Hops | **535**
- Internet Gateway Next Hop Euca2ools CLI | **536**
- NAT Instance Next Hop Euca2ools CLI | **536**
- Example: Creating a NAT Instance with Euca2ools CLI | **536**

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 82: 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

Table 82: Amazon VPC and OpenStack Contrail Feature Comparison *(Continued)*

Amazon VPC Feature	OpenStack Contrail Feature
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


                Assocation          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


                Assocation          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


                Assocation          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


                Assocation          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-00000003
```

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 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
```