# VMware NSX Advanced Load Balancer Monitoring and Operability Guide

VMware NSX Advanced Load Balancer 21.1.4

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

**VMware, Inc.**
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

# Contents

# Monitoring and Operability Guide Overview

1

The NSX Advanced Load Balancer uses capabilities to continuously monitor its various subsystems, to ensure High Availability(HA), Scalability, Security, and optimum resource utilization.

# NSX Advanced Load Balancer Monitoring Components

2

The monitoring methods used by the NSX Advanced Load Balancer include:

- Metrics
- Event Generation and Logging
- Notifications
- Monitoring of Controller and SEs
- Application Monitoring through Analytics
- Application Troubleshooting

This chapter includes the following topics:



## Metrics

Metrics are parameters or comparison standards used by the NSX Advanced Load Balancer to continually monitor subsystems. They are used to trigger alerts, notify administrators of important information, and are the building blocks of much of configuration automation.

Metrics apply to one or more of the following types of objects:

- Virtual Services
- Service Engines (SEs)
- Pools

## Types of Metrics

The **Analytics** view of the Controller application presents information regarding performance metrics. The following types of metrics are collected:

- **Virtual Service Metrics**:  Provides analytic data for client-side connections of the virtual service.
- **Pool Metrics**: Pool metrics provide data to assess and measure the performance of back end pool servers.
- **SE Metrics**: SE metrics include information on connection, bandwidth, transaction and memory usage for an SE.
- **Controller Metrics:** This includes CPU, memory, and disk usage information of all Controller nodes in a Cluster, tracked every minute.
- **User-defined Metrics:** Metrics can be customized to cater to specific needs of customers. Custom metric data can be obtained through API calls.
- **WAF Security Metrics:** Metrics analytics data obtained for intelligent Web Application Firewall (WAF) are included in WAF security metrics.

The following sections explain ways to access different metrics in detail, from the UI. For more information on the list of metrics available through API, see Accessing Metrics through API.

## Virtual Service Metrics

Virtual Service metrics provide analytic data for client side connections of the virtual service.

The **Analytics** view of a virtual service in the NSX Advanced Load Balancer presents information about the performance metrics of the virtual service. To access information on virtual service metrics:

1 Navigate to **Applications > Virtual Services**.

2 Click a Virtual Service to open the **Analytics** tab.

3 Click the **<** icon to open the **Virtual Service Metrics** tile.

**Metrics Tiles** continuously display time-averaged data. They govern what appears in the chart display. The format of the chart display on the **Analytics** tab changes according to the **Metric Tile** selected. The following metrics are available:

| Metric | Description |
|---|---|
| **End to End Timing** | The default metrics tile that causes end-to-end timing to be charted by default when the **Analytics** tab comes up. |
| **Throughput** | The total bandwidth passing through the virtual service, in Mbps. Pointing over this graph displays the throughput in Mbps for the selected time.<br>Throughput is measured as the number of bytes transferred between the client and the SE. This figure does not include data transferred between the SE and the servers. |
| **Open Conns** | The number of TCP client connections or UDP transactions currently in an open state. UDP transactions are counted even though they are technically connection-less. The number of open connections does not necessarily translate into the number of unique clients, as protocols such as HTTP 1.1 typically open six connections per client browser.<br>If the Maximum Concurrent Connections setting has been set for a virtual service, a horizontal red bar superimposed over the chart pane will show the maximum number of connections allowed. |
| **Conns** | Shows the average rate of connections completed per second. The tile also summarizes the percent of connections resulting in errors. The chart's mouse-over popup further breaks down the total by showing the rates of good, lossy and bad connections.<br>Lossy connections are those that experience issues such as re-transmissions, zero window size events, or high connection setup times. Bad connections are those which terminate ungracefully. |
| **Requests** | The number of responses to requests per second, which breaks down successful requests versus errors (such as 4xx or 5xx errors). Within the chart pane, toggling the radio buttons breaks down the errors by whether they are generated by the server or by the NSX Advanced Load Balancer .<br>For instance, if no servers are available in the pool, the NSX Advanced Load Balancer sends out an HTTP 500 status code. The **Client Logs** section provides more detail on specific errors. This metric is only available for HTTP virtual services. |
| **HTTP vs HTTP2 Requests** | Displayed in the form of requests-per-second rates. |

Virtual Service metrics are classified into Layer 4, Layer 7 and HTTP metrics.

Layer 4 Metrics

| Metric | Description |
|---|---|
| l4_client.apdexc | Measures the network connection quality (errors and lossy connections) between clients and the virtual service. |
| l4_client.apdexrtt | Measures the network connection quality against the Round Trip Time between clients and the virtual service. |
| l4_client.avg_application_dos_attacks | Number of application layer DoS attacks occurring. |
| l4_client.avg_bandwidth | Average transmit and receive network bandwidth between client and virtual service. |
| l4_client.avg_complete_conns | Rate of total connections completed (closed) per second. |
| l4_client.avg_connections_dropped | Rate of dropped connections per second. |
| l4_client.avg_dos_attacks | Number of application plus network layer DoS attacks occurring. |
| l4_client.avg_dos_bandwidth | Average transmit and receive network bandwidth between client and virtual service related to DoS attack. |
| l4_client.avg_errored_connections | Rate of connections per second classified as errored. |
| l4_client.avg_l4_client_latency | Average L4 connection duration which does not include client RTT. |
| l4_client.avg_lossy_connections | Rate of connections per second classified as lossy. |
| l4_client.avg_network_dos_attacks | Number of network layer DoS attacks occurring. |
| l4_client.avg_new_established_conns | New client connections per second. |
| l4_client.avg_policy_drops | Rate of connections dropped per second due to virtual service policies. This includes rate limits. |
| l4_client.avg_total_rtt | Average network round trip time between client and virtual service. |
| l4_client.max_open_conns | Maximum number of concurrently open connections. |
| l4_client.pct_application_dos_attacks | Percentage of HTTP application requests classified as a DoS attack. |
| l4_client.pct_connection_errors | Percentage of network connections to the virtual service classified as errored due to dropped or lossy. |
| l4_client.pct_connections_dos_attacks | Percentage of network connections classfied as a DoS attack. |
| l4_client.pct_dos_bandwidth | Percentage of network bandwidth classified as part of a DoS attack. |
| l4_client.pct_dos_rx_bytes | Percentage of bytes received that are classified as part of a DoS atack. |
| l4_client.pct_pkts_dos_attacks | Percentage of packets that are classified as part of a DoS attack. |
| l4_client.pct_policy_drops | Percentage of network connections dropped due to a virtual service network security policy. |

| Metric | Description |
|---|---|
| l4_client.sum_connection_errors | Total number of client network connections that were lossy or dropped. |
| l4_client.sum_finished_conns | Total number of completed connections. |
| l4_client.sum_lossy_connections | Total connections that were classified as lossy due to high packet retransmissions. |
| l4_client.sum_lossy_req | Total HTTP requests that were classified as lossy due to high packet retransmissions. |

## Layer 7 Metrics

| Metric | Description |
|---|---|
| l7_client.apdexr | Measures the clients view of the quality (response time and errors) of server responses. |
| l7_client.avg_application_response_time | Average server / application response latency. |
| l7_client.avg_blocking_time | Average time client was blocked as reported by client. Requires Client Insights set to Active. |
| l7_client.avg_browser_rendering_time | Average time browser spend rendering a web page. Requires Client Insights set to Active. |
| l7_client.avg_client_data_transfer_time | Average time required to transmit a file through layer 7 protocol (such as HTTP) from a virtual service to a client, excluding the Round Trip Time. |
| l7_client.avg_client_rtt | Average Round Trip Time between client and virtual service. |
| l7_client.avg_client_txn_latency | Client transaction time averaged across all HTTP requests.l7_client.avg_complete_responsesAverage rate of HTTP responses sent to clients per second. |
| l7_client.avg_connection_time | Average client connection latency as reported by client. Requires Client Insights set to Active to gather RUM data. |
| l7_client.avg_dns_lookup_time | Average DNS name lookup time as reported by the client. Requires Client Insights set to Active to gather RUM data. |
| l7_client.avg_dom_content_load_time | Average Dom content load time as reported by clients. Requires Client Insights set to Active to gather RUM data. |
| l7_client.avg_error_responses | Rate of HTTP error responses per second sent to clients. It does not include error codes that have been excluded in analytics profile. |
| l7_client.avg_frustrated_responses | Number of client HTTP requests that are completed but classified in the Frustrated latency bucket. |
| l7_client.avg_http_headers_bytes | Average size of HTTP headers per request. |
| l7_client.avg_http_headers_count | Average number of HTTP headers per request. |
| l7_client.avg_http_params_count | Average number of HTTP request parameters per request. |

| Metric | Description |
|---|---|
| l7_client.avg_page_download_time | Page download time as reported by clients. Requires Client Insights set to Active. |
| l7_client.avg_page_load_time | Page load time as reported by clients. Requires Client Insights set to Active. |
| l7_client.avg_params_per_req | Average number of HTTP request parameters per request, taking into account only requests with parameters. |
| l7_client.avg_post_bytes | Average size of HTTP POST request. |
| l7_client.avg_redirection_time | Latency incurred by following redirects as reported by clients. Requires Client Insights set to Active. |
| l7_client.avg_resp_1xx | Rate of HTTP 1xx responses sent to clients. |
| l7_client.avg_resp_2xx | Rate of HTTP 2xx responses sent to clients. |
| l7_client.avg_resp_3xx | Rate of HTTP 3xx responses sent to clients. |
| l7_client.avg_resp_4xx | Rate of HTTP 4xx responses sent to clients. |
| l7_client.avg_resp_4xx_avi_errors | Rate of HTTP 4xx responses sent to clients from the Controller, such as through custom security policies. This does not include errors excluded through the analytics profile or server generated 4xx errors. |
| l7_client.avg_resp_5xx | Rate of HTTP 5xx responses sent to clients. |
| l7_client.avg_resp_5xx_avi_errors | Rate of HTTP 5xx responses sent to clients from the Controller, such as through custom security policies. This does not include errors excluded through the analytics profile or server generated 5xx errors. |
| l7_client.avg_rum_client_data_transfer_time | Total client data transfer time as reported by clients. Requires Client Insights set to Active to gather RUM data. |
| l7_client.avg_satisfactory_responses | Number of client HTTP requests that are completed and classified in the Satisfied latency bucket. |
| l7_client.avg_server_rtt | Average Round Trip Time between the SE and server. |
| l7_client.avg_service_time | Average latency from the virtual services receipt of a request to start of the response. Requires Client Insights set to Active to gather RUM data. |
| l7_client.avg_ssl_connections | New SSL transactions per second (TPS) including SSL session reuse and failed handshake negotiations. |
| l7_client.avg_ssl_errors | SSL connection errors per second due to clients, protocol errors, network errors and handshake timeouts. |
| l7_client.avg_ssl_failed_connections | SSL connection errors per second due to protocol, network or timeout issues. |
| l7_client.avg_ssl_handshakes_new | New SSL transactions per second (TPS) excluding session reuse and errored connection attempts. |

| Metric | Description |
|---|---|
| l7_client.avg_ssl_handshakes_non_pfs | New SSL handshakes / transactions per second (TPS) that did not use Perfect Forward Secrecy. |
| l7_client.avg_ssl_handshakes_pfs | New SSL handshakes / transactions per second (TPS) that used Perfect Forward Secrecy. |
| l7_client.avg_ssl_handshakes_reused | Successfully resumed SSL sessions per second. |
| l7_client.avg_tolerated_responses | Number of HTTP requests which had response latency classified as Tolerated per the virtual service analytics profile. |
| l7_client.avg_total_requests | Client HTTP requests per second received by the virtual service. |
| l7_client.avg_uri_length | Average length of HTTP URI per request. |
| l7_client.avg_waf_disabled | Average number of transactions per second bypassing WAF. |
| l7_client.avg_waf_attacks | Average number of WAF attacks. |
| l7_client.avg_waiting_time | Average waiting time reported by the client. Requires Client Insights set to Active. |
| l7_client.max_ssl_open_sessions | Maximum number of concurrently open SSL sessions. |
| l7_client.pct_cache_hits | Percent of HTTP requests served from cache. |
| l7_client.pct_cacheable_hits | Percent of HTTP requests that were eligible to be served from cache. |
| l7_client.pct_get_reqs | Number of HTTP GET requests as a percentage of total requests received. |
| l7_client.pct_post_reqs | Number of HTTP POST requests as a percentage of total requests received. |
| l7_client.pct_response_errors | Percent of 4xx and 5xx HTTP responses. |
| l7_client.pct_ssl_failed_connections | Percent of SSL connection failures due to protocol, network or timeout errors. |
| l7_client.pct_waf_disabled | Transactions bypassing WAF as the percentage of total requests received. |
| l7_client.rum_apdexr | Quality (combination of performance and errors) of HTTP responses to clients based on RUM data. Requires Client Insights set to Active for RUM data |
| l7_client.sum_errors | Total number of HTTP 400 and 500 errors sent to a client. |
| l7_client.sum_get_reqs | Total number of HTTP GET requests. |
| l7_client.sum_http_headers_bytes | Total size of HTTP request headers in a given metrics interval. |
| l7_client.sum_http_headers_count | Total number of HTTP headers across all requests in a given metrics interval. |

| Metric | Description |
|---|---|
| l7_client.sum_http_params_count | Total number of HTTP request parameters. |
| l7_client.sum_num_rum_samples | Total number of samples used for RUM metrics. Requires Client Insights set to Active to gather RUM data. |
| l7_client.sum_other_reqs | Total number of HTTP requests that are not GET or POST requests. |
| l7_client.sum_post_bytes | Total size of HTTP POST requests. |
| l7_client.sum_post_reqs | Total number of HTTP POST requests. |
| l7_client.sum_reqs_with_params | Total number of HTTP requests containing at least one parameter. |
| l7_client.sum_total_responses | Total number of HTTP responses sent to clients. |
| l7_client.sum_uri_length | Total length of HTTP request URIs. |
| l7_client.sum_waf_disabled | Total number of requests bypassing WAF in a given metrics interval. |

## HTTP Metrics

| Metric | Description |
|---|---|
| http2_metrics.sum_get_reqs | Total number of HTTP2 GET requests. |
| http2_metrics.sum_post_reqs | Total number of HTTP2 POST requests. |
| http2_metrics.sum_other_reqs | Total number of HTTP2 requests that are not GET or POST requests. |
| http2_metrics.sum_resp_1xx | Total number of HTTP2 1xx responses. |
| http2_metrics.avg_resp_1xx | Rate of 1xx HTTP2 responses sent per second. |
| http2_metrics.sum_resp_2xx | Total number of HTTP2 2xx responses. |
| http2_metrics.avg_resp_2xx | Rate of 2xx HTTP2 responses sent per second. |
| http2_metrics.sum_resp_3xx | Total number of HTTP2 3xx responses. |
| http2_metrics.avg_resp_3xx | Rate of 3xx HTTP2 responses sent per second. |
| http2_metrics.sum_resp_4xx | Total number of HTTP2 4xx error responses. |
| http2_metrics.avg_resp_4xx | Rate of HTTP2 4xx responses sent per second. |
| http2_metrics.avg_resp_4xx_avi_errors | Rate of HTTP2 4xx responses as errors sent by NSX Advanced Load Balancer. It does not include any error codes excluded in the analytics profile and pool server errors. |
| http2_metrics.sum_resp_5xx | Total number of HTTP2 5xx error responses. |
| http2_metrics.avg_resp_5xx | Rate of 5xx HTTP2 responses sent per second. |

| Metric | Description |
|---|---|
| http2_metrics.avg_resp_5xx_avi_errors | Rate of HTTP2 5xx responses as errors sent by NSX Advanced Load Balancer. It does not include any error codes excluded in the analytics profile and pool server errors. |
| http2_metrics.avg_error_responses | Rate of HTTP2 error responses sent per second. It does not include errors excluded in analytics profile. |
| http2_metrics.pct_response_errors | Percent of HTTP2 4xx and 5xx responses. It does not include errors excluded in analytics profile. |
| http2_metrics.avg_total_requests | Average rate of client HTTP2 requests received by the virtual service per second. |
| http2_metrics.avg_complete_responses | Rate of HTTP2 responses sent per second. |

Insight Metrics

| Metric | Description |
|---|---|
| source_insights.avg_bandwidth | Rate of network throughput. |
| source_insights.avg_client_end2end_latency | Client end to end latency for successful responses. |
| source_insights.avg_complete_conns | Rate of connections per second. |
| source_insights.avg_complete_responses | Rate of completed responses including response errors. |
| source_insights.avg_dropped_conns | Rate of dropped connections per second |
| source_insights.avg_error_responses | Rate of errored requests per second. |
| source_insights.avg_http_timeout | Rate of HTTP timeouts per second. |
| source_insights.avg_policy_drops | Rate of connections per second dropped by policies. |
| source_insights.avg_total_requests | Rate of HTTP requests per second. |
| source_insights.max_open_conns | Maximum number of concurrent open connections. |

## Pool Metrics

Pool metrics provide data to assess and measure performance of back end pool servers.

The Analytics information available for a pool includes Metrics related to the Pool. To access the **Pool Metrics** page:

1 Navigate to **Applications** > **Pools**.

2 Click a pool to open the **Analytics** tab.

3 Click the **<** icon to open the **Pool Metrics** tile.

The **Analytics** tab under a pool instance presents information about various pool performance metrics. The Data shown is filtered by the period selected. Pool Metrics Tiles continuously display time-averaged data. They govern what appears in the chart display. The following metrics are available:

| Metric | Description |
|---|---|
| **End to End Timing** | The total time from the pool's **End to End Timing** graph. |
| **Throughput** | The total bandwidth passing through the virtual service in Mbps. Pointing over this graph displays the throughput in Mbps for the selected time. **Throughput** is measured as bytes transferred between the client and SE. It does not include data transferred between SE and the servers. |
| **Open Conns** | The number of TCP client connections or UDP transactions currently in an open state. UDP transactions are counted even though they are technically connection-less. The number of open connections does not necessarily translate into the number of unique clients, as protocols such as HTTP 1.1 typically open six connections per client browser.<br>If the **Maximum Concurrent Connections** setting has been set for a pool server, a horizontal red bar superimposed over the chart pane will show the maximum number of connections allowed. |
| **Estimated Capacity** | Estimated capacity of the pool in terms of maximum possible connections. |
| **Available Capacity** | Available capacity of the pool in terms of currently open connections. |
| **New Connections** | The number of client connections that were completed or closed over the selected period. The metric records the number of closed connections per second. |
| **Requests** | The number of HTTP requests sent to the servers assigned to the pool. This metric also shows errors sent to servers or returned by servers. Client requests that received an error response generated by the NSX Advanced Load Balancer (such as a 500 when no servers are available), are not forwarded to the pool or tracked in this view. |

| Metric | Description |
|---|---|
| **Servers** | Displays the number of servers in the pool and their health. The x-axis represents the number of HTTP requests or connections to the server. The y-axis represents the health score of the server. The chart enables viewing of servers in relation to their peers within the pool, thus helping to spot outliers.<br>Within the chart pane, click and drag the mouse over server dots to select and display a table of the highlighted servers below the chart pane. The table provides more details about these servers such as Server Name, IP Address, Port, Health, Ratio (the server's static load-balanced ratio), and Throughput. Clicking the name of a server shows the pool's **Server Insight** page, which shows additional health and resource status. |
| **CPU** | The average CPU usage during the time across all servers in the pool. |
| **Memory** | The average memory usage during the period across all servers in the pool. |

Pool Metrics are classified into Layer 4 and Layer 7 metrics.

Layer 4 Metrics

| Metric | Description |
|---|---|
| `l4_server.apdexc` | Measures the network connection quality (errors and lossy connections) between Service Engines and servers. |
| `l4_server.avg_available_capacity` | Estimated connections per second capacity available for a server. This metric is the difference between the max capacity and the average load on a server. |
| `l4_server.avg_bandwidth` | Average transmit and receive network bandwidth between client and virtual service. |
| `l4_server.avg_complete_conns` | Rate of new connections per second. |
| `l4_server.avg_connections_dropped` | Rate of dropped connections per second. |
| `l4_server.avg_errored_connections` | Rate of total error connections per second. |
| `l4_server.avg_est_capacity` | Estimated averaged capacity of a server's connections per second summed across all SEs. Pool level metric reflects summed estimated capacity across all the servers in the pool. |
| `l4_server.avg_goodput` | Application data goodput (data excluding network headers) as bytes per second between the SE and server. |
| `l4_server.avg_health_status` | Health score status of the server. 0 is down. |
| `l4_server.avg_lossy_connections` | Rate of lossy connections per second between the SE and server. |

| Metric | Description |
|---|---|
| l4_server.avg_new_established_conns | Rate of new established connections per second between SE and server. |
| l4_server.avg_open_conns | Number of concurrently open connections between SEs and servers. |
| l4_server.avg_pool_bandwidth | Transmit and receive network bandwidth between SEs and all servers in a pool. |
| l4_server.avg_pool_complete_conns | New connections per second across the virtual service or pool. |
| l4_server.avg_pool_errored_connections | Total connections classified as errored between Service Engines and all servers in a pool. |
| l4_server.avg_pool_new_established_conns | Total new connections per second established between Service Engines and all servers in a pool. |
| l4_server.avg_total_rtt | Average Round Trip Time across all completed (closed) connections. |
| l4_server.avg_uptime | Percent of time a server was marked as **up**. |
| l4_server.max_open_conns | Maximum number of concurrently open connections to a server. |
| l4_server.pct_connection_errors | Percent of network connections between Service Engines and a server that were dropped or lossy. |
| l4_server.pct_connection_saturation | Percent of a server's connection per second capacity that is estimated to be utilized. |
| l4_server.sum_connection_errors | Total number of network connections to a server that were dropped or were classified as lossy. |
| l4_server.sum_connections_dropped | Total number of network connections to a server that were dropped. |
| l4_server.sum_finished_conns | Total number of completed connections to a server. |
| l4_server.sum_health_check_failures | Total number of times a server was marked down by health monitors. |
| l4_server.sum_lossy_connections | Total number of network connections to a server that were classified as lossy. |
| l4_server.sum_lossy_req | Total number of HTTP requests that were classified as lossy due to high packet retransmissions. |

## Layer 7 Metrics

| Metric | Description |
|---|---|
| l7_server.apdexr | Quality (combination of performance and errors) of HTTP responses from servers to the virtual service. |
| l7_server.avg_application_response_time | Average response latency measured of pool servers. |
| l7_server.avg_complete_responses | Rate of server HTTP responses per second. |

| Metric | Description |
|---|---|
| l7_server.avg_error_responses | Rate of HTTP error responses sent per second. Does not include errors excluded in analytics profile. |
| l7_server.avg_frustrated_responses | Number of HTTP requests completed which had server response latency classified as Frustrated per the virtual service analytics profile. |
| l7_server.avg_resp_1xx | Rate of 1xx HTTP responses sent per second. |
| l7_server.avg_resp_2xx | Rate of 2xx HTTP responses sent per second. |
| l7_server.avg_resp_3xx | Rate of 3xx HTTP responses sent per second. |
| l7_server.avg_resp_4xx | Rate of 4xx HTTP responses sent per second. |
| l7_server.avg_resp_4xx_errors | Rate of 4xx HTTP responses per second minus error codes excluded by the analytics profile. |
| l7_server.avg_resp_5xx | Rate of 5xx HTTP responses sent per second. |
| l7_server.avg_resp_5xx_errors | Rate of 5xx HTTP responses per second minus error codes excluded by the analytics profile. |
| l7_server.avg_resp_latency | Latency measured for pool servers. |
| l7_server.avg_satisfactory_responses | Number of HTTP requests completed which had server response latency classified as Satisfied per the virtual service analytics profile. |
| l7_server.avg_tolerated_responses | Number of HTTP requests completed which had server response latency classified as Tolerated per the virtual service analytics profile. |
| l7_server.avg_total_requests | Rate of HTTP requests per second received by pool servers. |
| l7_server.pct_response_errors | Percent of HTTP 4xx and 5xx server responses. |
| l7_server.sum_get_reqs | Total number of HTTP GET requests received by servers. |
| l7_server.sum_other_reqs | Total number of HTTP requests that are not GET or POST request received by servers. |
| l7_server.sum_post_reqs | Total number of HTTP POST requests received by servers. |
| l7_server.sum_total_responses | Total number of HTTP responses sent from servers. |

## VMware Metrics

| Metric | Description |
|---|---|
| vm_stats.avg_cpu_usage | Percent of server CPU used. |
| vm_stats.avg_cpu_wait | Percent of time virtual machine was ready to run but could not due to CPU unavailable. This could be CPU limits configured in vCenter or other virtual machines stealing CPU time. |

| Metric | Description |
|---|---|
| vm_stats.avg_disk1_usage | Virtual disk1 capacity usage. |
| vm_stats.avg_disk2_usage | Virtual disk2 capacity usage. |
| vm_stats.avg_disk3_usage | Virtual disk3 capacity usage. |
| vm_stats.avg_disk4_usage | Virtual disk4 capacity usage. |
| vm_stats.avg_disk_commands_aborted | Rate of disk I/O commands that were ended prematurely in a virtual machine. |
| vm_stats.avg_disk_io | Rate of server disk reads plus writes per second. |
| vm_stats.avg_disk_read | Rate of data read from disk in kilobytes per second. |
| vm_stats.avg_disk_write | Rate of data written to disk in kilobytes per second. |
| vm_stats.avg_mem_swap_in | Total amount of data that has been read into machine memory from the swap file since the virtual machine was powered on. |
| vm_stats.avg_mem_swap_out | Total amount of data the VMkernel has written to the virtual machine swap file from machine memory. This statistic refers to VMkernel swapping and not to guest OS swapping. |
| vm_stats.avg_mem_usage | Percent of available server memory used. |
| vm_stats.avg_mem_vmmemctl | Virtual machine physical memory currently reclaimed from the virtual machine through ballooning. This is the amount of guest physical memory that has been allocated and pinned by the balloon driver. |
| vm_stats.avg_net_dropped | Rate of dropped received and transmit packets. It be an indication of network congestion. |
| vm_stats.avg_net_usage | Transmit plus receive network bandwidth for the virtual machine. |
| vm_stats.avg_port_usage | Percent of high ports used. |
| vm_stats.avg_uptime | Percent of time the virtual machine was up during a time interval. |
| vm_stats.avg_virtual_disk_commands_aborted | Number of SCSI commands that were ended prematurely. |

**Note** The VMware metrics listed for Pool Metrics are also part of SE Metrics.

## SE Metrics

SE metrics includes information on connection, bandwidth, transaction and memory usage for an SE.

The Analytics information available for an SE includes performance metrics for the SE over a defined time. To access the **SE Metrics** page:

1 Navigate to **Infrastructure > Cloud Resources > Service Engine** .

2 Select a cloud from the **Select Cloud**drop-down menu.

3 Click the name of an SE to open the **Analytics** tab for the SE.

4 Click the **<** icon to open the **Service Engine Metrics** tile.

The **Analytics** tab of the SE presents performance metrics information for the SE. Data shown is filtered by the time selected. **Service Engine Metrics** Tiles continuously display time-averaged data. They govern what appears in the chart display. The following metrics are available:

| Metric | Description |
| --- | --- |
| **Throughput** | Total bandwidth flowing through the SE for all virtual services being hosted by that SE. It includes the bandwidth flowing in and out of the SE between the client and the virtual service, and the traffic between the SE and the servers. An SE can report approximately double the throughput of its virtual services. |
| **CPU Usage** | Displays the utilization of the CPUs allocated to the SE. The total number of CPUs appears in the SE **Quick Info Popup** window. Under normal conditions, CPU usage should not regularly exceed 90%, as this may cause latency in the virtual services and disrupt the client experience.<br>The **CPU Usage** metric tile shows a horizontal bar indicating current usage, with a red line at the right to indicate how close the SE is to pushing the limits of its available CPU capacity. The CPU usage can be indirectly controlled or improved by taking actions, such as:<br>**Configuration**: Changing the configuration of virtual services, such as changing SSL or compression settings, impacts the CPU usage.<br>**CPU Allocation**: Allocating more vCPUs per SE. The default setting is two vCPUs per SE. Increasing this number is particularly useful for tasks such as SSL termination or compression, which heavily consume CPU resources. The setting for the number of vCPUs assigned to an SE is in the SE group.<br>**Scale Out:**Reduce the CPU load by scaling the SE's virtual services across additional SEs. This increases the total capacity and reduces the load on this SE. The high availability setting of the SE group dictates when a virtual service should be scaled out across additional SEs or simply migrated away from a busy SE.<br>**CPU Reservation:** By default, CPUs resource is not reserved in a VMware deployment. Within vCenter, you can enable reservation for the SE's virtual machine. This guarantees that other virtual machines sharing the same physical host server are not able to borrow or compete for CPU resources. This setting can be changed in the SE group properties. Changes will apply for only new SEs. Changes for the existing SEs have to be done manually within vCenter. |
| **Memory Usage** | Amount of used versus available memory. Memory utilization should not exceed 90% for an extended period. |
| **Rx Packets** | The received network packets for the SE. |
| **Tx Packets** | The number of packets successfully transmitted for the SE. |

| Metric | Description |
|---|---|
| **Interface Throughput** | The combined throughput for all network interfaces utilized by this SE. Throughput is measured as both client and server side of any virtual services, plus the management traffic between the SE and the Controllers. |
| **Virtual Service Throughput** | The combined throughput for all network interfaces utilized by this SE. Throughput is measured as both client and server side of any virtual services, plus the management traffic between the SE and the Controllers. |
| **Connection Memory Usage** | Percentage of **connection memory**used by the SE. |
| **Dynamic Memory Usage** | Percentage of **dynamic memory** used by the SE. |
| **SSL Cache Usage** | Percentage of SSL cache memory used by the SE. |
| **Persistent Table Usage** | Persistent Table used by the SE expressed as %. Since client IP persistence is stored locally on each SE, larger tables consume more memory. |
| **Buffer Usage** | Percentage of **buffer memory**used by the SE. |

The list of SE metrics available in the NSX Advanced Load Balancer system are listed in the table below:

| Metric | Description |
|---|---|
| `se_if.avg_bandwidth` | Transmit and receive network bandwidth across all SE interfaces. |
| `se_stats.avg_connection_mem_usage` | Percent of connection memory consumed. |
| `se_stats.avg_connections` | Rate of client network connection attempts (SYNs) per second across all virtual services for an SE. |
| `se_stats.avg_connections_dropped` | Number of connections dropped or failed to establish across all virtual services for an SE. Excludes drops due to a policy action. |
| `se_stats.avg_cpu_usage` | Hosts view of the Service Engines actively utilized CPU usage as a percent of total available CPU. |
| `se_stats.avg_disk1_usage` | Percent of SE disk 1 capacity used. |
| `se_if.avg_eth0_bandwidth` | Transmit and receive network bandwidth for an SE interface. |
| `se_if.avg_eth10_bandwidth` | Transmit and receive network bandwidth for an SE interface. |
| `se_if.avg_eth11_bandwidth` | Transmit and receive network bandwidth for an SE interface. |
| `se_if.avg_eth12_bandwidth` | Transmit and receive network bandwidth for an SE interface. |
| `se_if.avg_eth13_bandwidth` | Transmit and receive network bandwidth for an SE interface. |

| Metric | Description |
| --- | --- |
| se_if.avg_eth14_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth15_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth16_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth17_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth18_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth19_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth1_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth20_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth21_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth22_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth23_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth2_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth3_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth4_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth5_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth6_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth7_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth8_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_if.avg_eth9_bandwidth | Transmit and receive network bandwidth for an SE interface. |
| se_stats.avg_mem_usage | Percent of allocated memory used. |
| se_stats.avg_packet_buffer_header_usage | Percent of all network packet buffers used. |
| se_stats.avg_packet_buffer_large_usage | Percent of large network packet buffers used. |

| Metric | Description |
|---|---|
| se_stats.avg_packet_buffer_small_usage | Percent of small network packet buffers used. |
| se_stats.avg_packet_buffer_usage | Percent of total configured network packet buffers used. |
| se_stats.avg_persistent_table_usage | Percent of session persistence table used. |
| se_if.avg_rx_bandwidth | Received network bandwidth across all interfaces for an SE. |
| se_if.avg_rx_bytes_dropped | Bytes per second of received packets that were dropped. Includes packets across all SE interfaces. |
| se_if.avg_rx_pkts | Received network packets across all SE interfaces. |
| se_if.avg_rx_pkts_dropped | Received packets dropped per second. Includes packet drops across all interfaces. |
| se_stats.avg_ssl_session_cache_usage | Percent of SSL session cache used. |
| se_if.avg_tx_pkts | Network packets transmitted across all SE interfaces. |
| se_if.max_se_bandwidth | Maximum bandwidth through an SE within the sample period. |
| se_stats.pct_connections_dropped | Percent of SE connections dropped. |
| se_if.pct_rx_bytes_dropped | Percent of SE bytes dropped. |
| se_if.pct_rx_pkts_dropped | Percent of SE packets dropped. |
| se_stats.pct_syn_cache_usage | Percent of SYN cache used. Higher usage indicates too many incomplete open connection attempts. |
| se_stats.avg_dynamic_mem_usage | Percent of average dynamic memory used. |

### VMWare Metrics

For more information on VMware metrics related to SE, see VMware Metrics.

## Controller Metrics

The NSX Advanced Load Balancer system monitors Controller nodes in a Cluster. Analytics information about CPU, memory, and disk usage across Controller nodes are collected.

### Background

This section explains troubleshooting crash, restart, and similar issues on the NSX Advanced Load Balancer Controller, using REST API. To troubleshoot these issues, it is essential to know the processes that were running before and after a particular incident. Troubleshooting license issues also requires various metrics associated with the Controller and Service Engines(SEs).

The NSX Advanced Load Balancer Controller collects two types of analytics data. They are:

1 **Controller Metrics**: This includes CPU, memory, and disk usage information of all Controller nodes in a Cluster, tracked every minute.

2 **Process Metrics**: This includes CPU, memory and disk usage, the number of context switches, swap usage, IO-read bytes, IO-write bytes, the number of threads, and number of files opened for each process running on Controller nodes.

## Instructions

The data for controller metrics and process metrics are captured every minute and sent to the metrics manager process running on the Controller. The metric manager performs five-minute aggregations and writes into the database. These metrics are thus available to authorized users through the REST API or SDK.

## Example: Example - Finding the vm_uuid

Use the following statment to find the vm_uuid for the desired Controller node:

```
https://10.10.24.102/api/cluster
```

In the above REST API call, 10.10.24.102 is the Controller IP. Below is the output of the REST API call:

```
{
  "nodes" : [
    {
      "ip" : {
       "type" : "V4",
       "addr" : "10.10.24.102"
    },
    "vm_hostname" : "node1.controller.local",
    "vm_uuid" : "005056b015e3",
    "name" : "10.10.24.102",
    "vm_mor: "vm_147457"
    }
],
  "tenant_uuid" : "admin",
  "uuid" : "cluster-63087542-5f3b-44ec-8249-bf6428bed78f",
  "name" : "cluster-0-1"
}
```

In the output, 005056b015e3 is the `vm_uuid` for the VM name or the Controller IP 10.10.24.102. Use this `vm_uuid` for the REST API calls to collect Controller metrics and process metrics.

## Example: Collecting Controller metrics

Use the following REST API call to collect Controller metrics.

```
/api/analytics/metrics/controller/<vm-uuid>/?
metric_id=controller_stats.avg_cpu_usage&pad_missing_data=false&limit=3&step=300
```

Replace **vm_uuid** with the **vm_uuid_output** of the Controller node.

## Example: Fetching process metrics

- Use the following API call to collect data and logs for all processes running on a Controller node:

```
/api/analytics/metrics/controller/<vm-uuid>/?
metric_id=process_stats.avg_fds&pad_missing_data=false&limit=3&step=300&ob
j_id=*
```

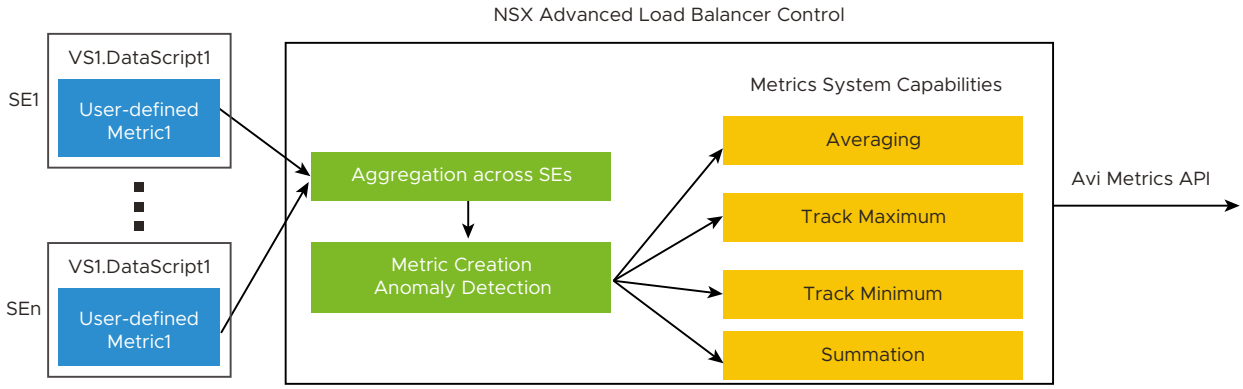
- Use the following API call for a particular process named avi-health with PID (process ID) 14257:`/api/analytics/metrics/controller/<vm-uuid>/?metric_id=process_stats.avg_fds&pad_missing_data=false&limit=3&step=300&obj_id=avi-health:14257`

**Note** For collecting metrics for a particular process, provide the value of the process name and the process ID, instead of just * for obj_id as shown in the above REST API calls.

## User-defined Metrics

The NSX Advanced Load Balancer tracks and maintains metrics data based on several parameters. In some cases, organizations might need additional data, unique to their applications or environment. To satisfy this need, user-defined Metrics are provided.

Three new DataScript functions interact with the NSX Advanced Load Balancer metrics subsystem to enable organizations to create, manipulate, and access their custom-defined metrics. REST API calls can be used to incorporate the custom data into the customer's monitoring systems.




A user-defined metric is:

- An integer and can never be negative.
- Created and subsequently manipulated by a DataScript running in the SEs upon which the virtual service has been placed.
- Collected and aggregated by the Controller(s). The Controller can detect anomalies in the values of these metrics.
- Stored over time so that various metrics system capabilities like averaging, min/max tracking, and summation can be applied.
- Able to be queried through the Metrics API.

Any given user-defined metric is scoped to the virtual service that creates it through a DataScript. Since a DataScript can be used by more than one virtual service, the Controller has to keep identically named metrics in different virtual services isolated from one another. The interpretation of a user-defined metric is left completely up to the user. Its value is based on conditions detected within the DataScript in which the metric is referenced.

When a metric crosses some user-defined threshold, any number of actions or no action can be triggered within the DataScript.

## Example: User-defined Metrics using DataScripts

1 Use the `avi.vs.log()` DataScript function to simply report the value of the metric.

2 Use the `avi.vs.rate_limit( type, string_to_limit, [defer_action=False] )` DataScript function to rate-limit the virtual service.

**Note** In most cases, metric data is extracted through calls to the REST API than through DataScripts.

## User-defined Metric Types

The NSX Advanced Load Balancer supports two types of user-defined metrics:

- **avi.vs.analytics.METRICTYPE_COUNTER**: A metric of this type can be incremented or cleared.
- **avi.vs.analytics.METRICTYPE_GAUGE**: A metric of this type can be incremented, decremented, cleared, or set.

## DataScript Functions for User-Defined Metrics

The DataScript functions that manipulate the above metrics are:

- `avi.vs.analytics.counter(metric_name, [operation], [value])`: Increments or clears the named counter metric.
- `avi.vs.analytics.gauge(metric_name, [operation], [value])`: Increments, decrements, sets or clears the named gauge metric.
- `avi.vs.analytics.get_metric(metric_name, metric_type)`: Returns the value of the metric, be it of type counter or gauge. Before doing so, the function first checks if metric_name has been defined. If it has, it then compares the value of metric_type to the type previously stored. If metric_name has never been defined, or the type is not correct, the function reports an error on the log, and the request/response causing this function call is canceled.

## User-Defined Metric Lifecycle

User-defined metrics are always aged and automatically destroyed if not used for more than two hours. Every time a counter or gauge is read or updated, its age is refreshed. Aging provides automatic garbage collection of user-defined metrics.

## Accessing User-Defined Metrics

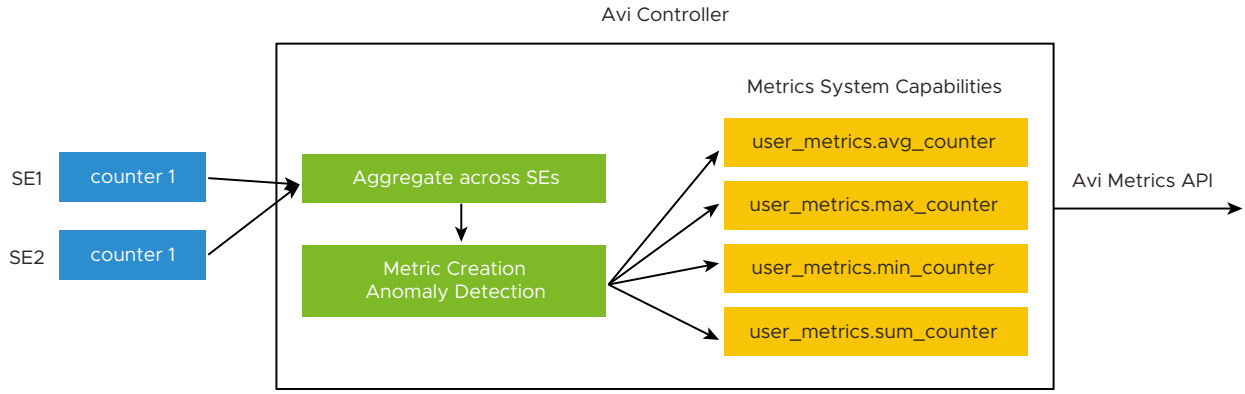
The NSX Advanced Load Balancer Controller provides access to these metrics through REST API calls.

In a call such as the following:

```
api/analytics/metrics/virtualservice/<vs_uuid>?
metric_id=user_metrics.sum_counter&obj_id=<obj_id>&step=300&limit=1
```

obj_id is either the ID of the counter or '*' to identify all the active counter metrics.

## Aggregated values that can be queried for avi.vs.analytics.METRICTYPE_COUNTER




- user_metrics.sum_counter: Sum of counter metric values reported over a given period.

For example: The latest 5-minute sample of aggregated user metric having a **counter_id** equal to "Foo" can be fetched using following API code:

```
api/analytics/metrics/virtualservice/<vs_uuid>?
metric_id=user.sum_counter&obj_id=Foo&step=300&limit=1
```

- **user_metrics.avg_counter**: The time averaged **per-second**values of **counter metric** over a given period.

For example: The latest 5-minute sample of averaged **per-second** rate of **counter metric** "Foo" can be requested using the following API call:

```
api/analytics/metrics/virtualservice/<vs_uuid>?
metric_id=user.avg_counter&obj_id=Foo&step=300&limit=1
```

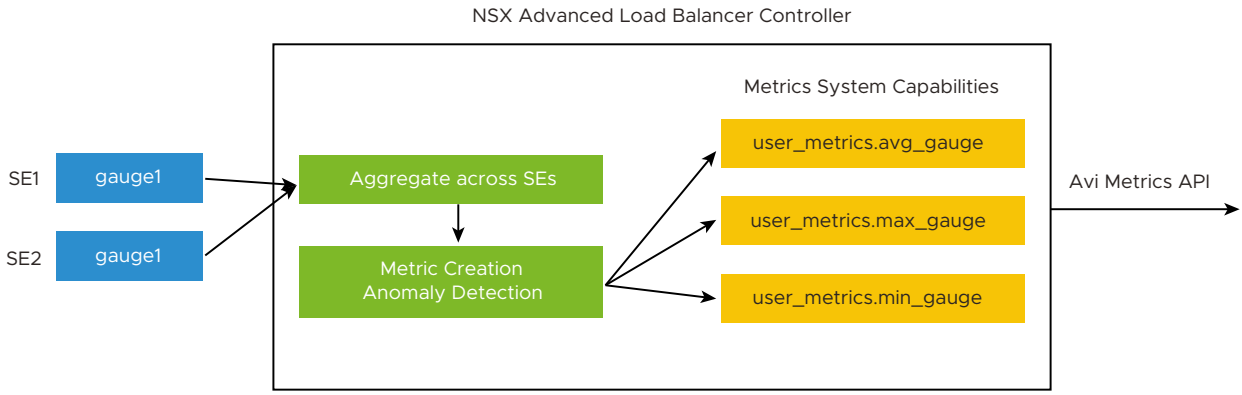
- **user_metrics.max_counter** and **user_metrics.min_counter**: Maximum/minimum per-second reported values of counter metric Foo over a given period.

For example: the **per-secondmax** and **min** values can be requested using these API calls:

```
api/analytics/metrics/virtualservice/<vs_uuid>?
metric_id=user.max_counter&obj_id=Foo&step=300&limit=1
```

```
api/analytics/metrics/virtualservice/<vs_uuid>?
metric_id=user.min_counter&obj_id=Foo&step=300&limit=1
```

### Aggregated values that can be queried for avi.vs.analytics.METRICTYPE_GAUGE




- **user_metrics.avg_gauge**: Average of raw counter value of type gauge reported over given period.

For example: fetch counter 'foo' through API

```
/api/analytics/metrics/virtualservice/?
metric_id=user_metrics.avg_gauge&limit=1&step=300&obj_id=foo
```

- user_metrics,max_gauge: Maximum raw counter value of type gauge reported over given period.

For example: fetch counter 'foo' through API.

```
/api/analytics/metrics/virtualservice/?
metric_id=user_metrics.max_gauge&limit=1&step=300&obj_id=foo
```

- **user_metrics.min_gauge**: Minimum raw counter value of type gauge reported over given period.

For example: fetch counter 'foo' through API.

```
/api/analytics/metrics/virtualservice/?
metric_id=user_metrics.min_gauge&limit=1&step=300&obj_id=foo
```

## WAF - Security Metrics

Metrics analytics data obtained for intelligent Web Application Firewall (WAF) are available for viewing in the Controller UI.

To view WAF related metrics:

1 Navigate to **Applications > Virtual Services**.

2 Click the virtual service mapped to the WAF policy, and navigate to the **WAF** tab.

The chart in this tab displays WAF rule hits against the chosen time frame. This helps analyze denied requests and their corresponding trigger.

The following fields show specific hit counts for each listed element:

- Group
- Rule
- Tag
- Client IP
- Path
- Match Element

Elements in each field are displayed with the corresponding hit count. On discovering a false positive, any rule or group can be disabled using the toggle button. Clicking on any element in each field creates a specific filter. The field **Popular Combinations** displays the known combinations and their hit counts related to the chosen filter. The filter can be reset by clicking Reset filters.

# Retrieving metrics through APIs

Metrics are supported on **virtualservice**, **serviceengine**, **pool**, **server**, **virtualmachine**, and **host** objects.

All metrics API have the following syntax:

```
https://${controller}/api/analytics/metrics/ ${object_type}/${object_uuid}/?
metric_id=${metric_id}& ${additional_kv_options}
```

By default, APIs use tenant admin. In order to perform queries for a specific non admin tenant, add HTTP header **X-AVI-TENANT.** When the **X-AVI-TENANT** header is specified, the user access is checked against that tenant before analytics APIs are served.

For example, use the following API to query `metricsl4_client.avg_bandwidth`, `l7_client.avg_complete_responses` for the virtual service `virtualservice-abc` at intervals of five minutes and for the past 6 hours (72 samples):

```
https://avicontroller/api/analytics/metrics/ virtualservice/
virtualservice-abc/?metric_id=l4_client.avg_bandwidth,
l7_client.avg_complete_responses&limit=72&step=300
```

Of all the options, **limit**, **step**, **start** and **stop** work together. **step** denotes the duration granularity of the metrics aggregation. For example, **step**=300 means the **average**, **sum**, **max** etc. are computed using metrics in a duration window of five minutes.

The metrics API returns one value every five minutes as denoted by **step**. The NSX Advanced Load Balancer solution supports **step** up to 300,3600,86400 values. When **start** and **stop** are not specified, **stop** is assumed to be the latest time in UTC and **start** is computed using the formula: **stop** - **limit** * **step**. When only **start** and **step** are specified, the solution computes the limit based on the formula: (stop - start) / step, to return all the metrics available starting from **time** = **start**.

For more information on retrieving metrics through API, see Metrics API.

For information on configuration steps for metrics collection, see Metrics Collection.

### Example: Using API for retrieving metrics

1. Get metrics for last 6hrs: limit=72 & step=300.

2. Get metrics for last 1day: limit=288 & step=300.

3. Get metrics since T1: start=T1 & step=300.

4. Get latest single metric: limit=1 & step=300.

## Using Metrics for Push Notifications

Push notifications allow stakeholders to be notified of important activities of the Controller, even when they are not logged into the NSX Advanced Load Balancer application. The notifications can be sent through mediums like email, external logs, etc., and can be configured to be triggered based on evaluation of metrics parameters or changes in their values.

Alert Configurations are used to define triggers that will generate an alert. For a detailed explanation on alert configurations, see Alert Config.

For more information on evaluating metrics, see Controller Metrics and User-defined Metrics.

## Events

Events are used throughout the NSX Advanced Load Balancer application to provide a history of relevant changes that have occurred. Events are permanent records. Alerts are configured to get triggered based on the occurrence of an event.

### Events Overview

In the UI, events are viewed within the context of specific objects, such as a virtual service, a pool, or a server. In contrast, viewing events from the **Operations** menu provides an unfiltered view of all events across the system or the tenant. It also provides useful information for system debugging.

For a detailed list of events monitored by the NSX Advanced Load Balancer, see Events List

Apart from logging significant incidents within the NSX Advanced Load Balancer, events are used to:

- Trigger alerts for providing external notification.

- Potentially scale application capacity.
- Alter the NSX Advanced Load Balancer configuration.

For more information, see Alerts Overview.

## All Events

Navigate to **Operations > Events > All Events** to view all the events in the entire NSX Advanced Load Balancer system. The other **Events** pages in the system display only the events relevant to the context. For example, the **Events** page under a virtual service only shows events relevant to that virtual service.

The input fields of the Events page are detailed in Table 1:

Table 1: Events Search

| Field | Description |
|---|---|
| **Search** | The **Search** field allows you to filter the events using whole words contained within the individual events. |
| **Refresh** | Updates the events that are displayed for the selected time frame. |
| **Number** | Total number of entries displayed. The date/time range of those events appear below the **Search** field. |
| **Clear Selected** | If filters have been added to the Search field, the X icon on the right side of the search bar will remove those filters and clear selection. Each active search filter will also contain an X that can be clicked to remove a specific filter. |
| **Histogram** | The Histogram shows the number of events over a selected time per. The X-axis represents time, while the Y-axis shows the number of events that occurred during a period, represented as a bar. |
| **Include Internal** | By default, a number of events are not shown as they tend to be noisy and less relevant for general purpose. The NSX Advanced Load Balancer support may ask to enable this option to troubleshoot more obscure issues. |

The events that matched the input are displayed in the results. The information displayed for each event in the result is detailed in Table 2.

Table 2: Events

| Field | Description |
|---|---|
| **Timestamp** | Date and time the event occurred. Highlighting a section of the Histogram allows filtering of events within a smaller time window. |
| **Event Type** | This page is scoped to only show Configuration event types. Configuration events track changes to the NSX Advanced Load Balancer configuration. These changes may be made by an administrator (through the CLI, API, or GUI), or by automated policies. |
| **Resource Name** | Name of the object related to the event. For User_Login Events, this shows the **username** of the user that attempted to log in. |
| **Event Code** | A short event definition such as User_Login, Config_Create, Config_Modify, or Config_Delete |
| **User** | The name of the user that has performed the activity for which the event has been generated. |

| Field | Description |
|---|---|
| **Description** | A complete event definition. For configuration events, the description includes name of the account user that made the change. |
| **Expand/Contract** | Clicking the + plus or - minus sign for an event log either expands that event log to display more detail or contracts that event log to display only summary information.<br>For configuration update events, expanding the event log highlights differences between the previous and updated configurations. The updates are color coded to indicate the type of update:<br>■ Green for new fields<br>■ Red for removed fields<br>■ Yellow for changed fields |

## Types of Events

Based on the source of occurrence or context, events can be classified as:

- Controller Events
- Config Audit Trail
- Service Engine Events
- Virtual Service Events
- Security Events

## Controller Events

The NSX Advanced Load Balancer UI maintains a view for events specifically generated for the controller.

The **Administration > Controller > Events** page lists Controller-related events.

The Search Input and Events display format for the **Controller Events** tab is same as the **All Events** page. For more information, see Table 1: Events Search and Table 2: Events.

## Config Audit Trail

The **Config Audit Trail** page presents an audit trail of user activity events and changes to the system configuration over the selected period. This is a filtered view of All Events, showing only events related to configuration changes.

Navigate to **Operations > Events > Config Audit Trail**. The table at the bottom of the **Config Audit Trail** tab displays the events matching the current time window and any potential filters.

The Search Input and Events display format for the **Config Audit Trail** page is same as the **All Events** page. For more information, see Table 1: Events Search and Table 2: Events.

## Service Engine Events

The NSX Advanced Load Balancer system monitors the activities of the SE and generates system events over a selected period. Such events are filtered for display based on the context in which they are viewed.

When viewing events for an SE, only events that are relevant to the SE object are displayed.

Navigate to **Infrastructure > Cloud Resources > Service Engine**. Click an SE. The **Events** tab displays the events that matched the current time window and any potential filters.

The Search Input and Events display format for the **Service Engine Events** tab is the same as the **All Events** page. For more information, see Table 1: Events Search and Table 2: Events.

## Virtual Service Events

Virtual Services are monitored by the NSX Advanced Load Balancer and system events are generated. The events are used for reactive troubleshooting and proactively triggering alerts to the user. These events are filtered for display based on the context in which they are viewed.

Navigate to **Application > Virtual Services**. Click a Virtual Service. The **Events** tab displays the events that matched the current time window and any potential filters.

The Search Input and Events display format for the **Virtual Service Events** tab is the same as the **All Events** page. For more information, see Table 1: Events Search and Table 2: Events.

## Security Events

The NSX Advanced Load Balancer continually assesses the health of each virtual service. This health information is available for viewing in both summary and detailed form.

Each virtual service has a health score, which shows the virtual service health as both a color code and a set of numeric scores. The final health score is comprised of a positive performance score and three penalties.

The security penalty provides insight into a current security related issue (such as a current DoS attack) or potential risk (such as SSL configuration which leaves the site vulnerable to the POODLE attack).

Ideally, the security penalty must be **zero**, which means that it is not detracting from the health or risk of a virtual service. A **non-zero** security penalty can be due to an issue with SSL or a DDoS attack event. The following section explores the components that could generate a security penalty.

### View Security Insights

To view a security insights for a virtual service:

1 Click the icon of the virtual service shown on the Dashboard.

2 Click the virtual service name.

3 Click Security on the menu bar.

Detailed security information for the virtual service is shown in two panes:

- SSL information (left pane)
- DDoS information (right pane)

Security Insights for the virtual service are organized into the following main categories:

- **SSL Distribution**: The SSL section in the default security page shows details of the most relevant SSL data about client connections terminated on SEs within the selected period of time. If SSL termination is not performed on the virtual service, this section will have no data.
- **SSL Score**: The SSL Score section in the default security page shows details of major factors affecting the SSL Score.
- **DDoS**: The DDoS section in the default security page breaks down distributed denial of service data for the virtual service into the most relevant layer 4 and layer 7 attack data.

For more information on DDOS attack mitigation, Rate Limiters and Datascripts, see #unique_20.

## Events List

The NSX Advanced Load Balancer continuously adds new events to its system to improve its monitoring capabilities. The events are used to trigger alerts, notify administrators of important information, and are the building blocks of the configuration automation.

### Events and Description

| Event | Description |
|---|---|
| ADD_NW_FAIL | Failed to add network(s) to SE for virtual service and the reason. |
| ADD_NW_SE | Network(s) added for the virtual service. |
| ANOMALY | Anomalous metric of value and deviation. |
| APIC_ATTACH_CIF_TO_LIF | Attached CIF to LIF. |
| APIC_BAD_CREDENTIALS | Bad user credentials when logging into APIC Controller. |
| APIC_BIND_VNIC_TO_NETWORK | Bind vNICs to network. |
| APIC_CREATE_CDEV | Created APIC CDEV. |
| APIC_CREATE_LIF_CONTEXTS | Created APIC LIF contexts contract_graphs. |
| APIC_CREATE_LIFS | Created APIC LIFs. |
| APIC_CREATE_NETWORK | Created networks. |
| APIC_CREATE_TENANT | Created tenant. |
| APIC_DELETE_CDEV | Deleted APIC CDEV. |
| APIC_DELETE_LIF_CONTEXTS | Deleted APIC LIF contexts contract_graphs. |
| APIC_DELETE_LIFS | Deleted APIC LIFs. |
| APIC_DELETE_NETWORK | Deleted networks. |
| APIC_DELETE_TENANT | Deleted tenant. |

| Event | Description |
|---|---|
| APIC_DETACH_CIF_FROM_LIF | Detached CIF from LIFs. |
| APIC_VS_PLACEMENT | Place virtual service. |
| AWS_ACCESS_FAILURE | AWS access/setup failure. |
| AWS_ACCESS_SUCCESS | AWS access/setup succeeded. |
| AWS_IMAGE_UPLOAD_FAILURE | AWS image upload failure. |
| AWS_IMAGE_UPLOAD_SUCCESS | AWS image upload succeeded. |
| CACHE_OBJ_ALLOC_FAIL | HTTP cacheable objects dropped due to memory allocation failure. |
| CC_CLUSTER_VIP_CONFIG_FAILURE | Cluster VIP configuration in cloud failed. |
| CC_CLUSTER_VIP_CONFIG_SUCCESS | Cluster VIP configuration in cloud succeeded. |
| CC_CLUSTER_VIP_DECONFIG_FAILURE | Cluster VIP deconfiguration in cloud failed. |
| CC_CLUSTER_VIP_DECONFIG_SUCCESS | Cluster VIP deconfiguration in cloud succeeded. |
| CC_CONFIG_FAILURE | Cloud configuration failed. |
| CC_DECONFIG_FAILURE | Cloud deconfiguration failed. |
| CC_DELETE_VIP_FAILURE | VS delete failed. |
| CC_GENERIC_FAILURE | Cloud generic failure. |
| CC_HEALTH_FAILURE | Cloud health check failed. |
| CC_HEALTH_OK | Cloud health check passed. |
| CC_IP_ATTACH_FAILURE | IP attach to port failed. |
| CC_IP_ATTACHED | IP address attached to port. |
| CC_IP_DETACH_FAILURE | IP detach failure. |
| CC_IP_DETACHED | IP address detached from port. |
| CC_MARATHON_SERVICE_PORT_ALREADY_IN_USE | Application requested a service port already in use. |
| CC_MARATHON_SERVICE_PORT_OUTSIDE_VALID_RANGE | Application requested a service port outside the range of its Marathon instance. |
| CC_SE_CREATED | SE created. |
| CC_SE_CREATION_FAILURE | SE creation failure. |
| CC_SE_DELETED | SE deleted. |
| CC_SE_DELETION_DETECTED | SE deleted from cloud. |
| CC_SE_DELETION_FAILURE | SE delete failed. |

| Event | Description |
|---|---|
| CC_SYNC_SERVICES_FAILURE | Sync services failed. |
| CC_SYNC_SERVICES_SUCCESS | Sync services success. |
| CC_TENANT_INIT_FAILURE | Initialization of tenant in cloud failed. |
| CC_UPDATE_VIP_FAILURE | VS create/update failed. |
| CC_VIP_DNS_REGISTER_FAILURE | DNS registration of VS in cloud failed. |
| CC_VNIC_ADDED | New vNIC added to SE. |
| CC_VNIC_ADDITION_FAILURE | vNIC addition failure. |
| CC_VNIC_DELETED | vNIC deleted from SE. |
| CC_VNIC_DELETION_FAILURE | vNIC delete failed. |
| CONFIG_ACTION | Action executed by. |
| CONFIG_CREATE | Config create status is (performed by user). |
| CONFIG_DELETE | Config delete status is (performed by). |
| CONFIG_INTERNAL_CREATE | Config created internally. |
| CONFIG_INTERNAL_UPDATE | Config updated internally. |
| CONFIG_UPDATE | Config update status is (performed by user). |
| CONN_DROP_MAX_FLOW_TBL | connection(s) dropped due to flow table limit. |
| CONN_DROP_MAX_PERSIST_TBL | connection(s) dropped due to persistence table limit. |
| CONN_DROP_MAX_SYN_TBL | connection(s) dropped due to SYN table limit. |
| CONN_DROP_NO_CONN_MEM | connection(s) dropped due to insufficient memory. |
| CONN_DROP_NO_PKT_BUFF | connection(s) dropped due to low packet buffer. |
| CONN_DROP_POOL_LB_FAILURE | pool load balancing decision(s) failed. |
| CONTROLLER_CPU_HIGH | CPU usage (aggregated across all cores) of the Controller is over 85%. |
| CONTROLLER_DISK_HIGH | Disk usage of the Controller is over 85%. |
| CONTROLLER_LEADER_FAILOVER | Controller cluster leader failover. |
| CONTROLLER_MEM_HIGH | Memory usage of the Controller is over 85% (18.2.5+). |
| CONTROLLER_NODE_DB_REPLICATION_FAILED | Controller node database replication failed. |
| CONTROLLER_NODE_JOINED | Controller node joined the Controller cluster. |
| CONTROLLER_NODE_LEFT | Controller node left the Controller cluster. |

| Event | Description |
|---|---|
| CONTROLLER_NODE_STARTED | Controller node started. |
| CONTROLLER_SERVICE_FAILURE | Controller service failed. |
| CONTROLLER_WARM_REBOOT | Controller warm restarted. |
| CREATE_SE_FAIL | SE creation failed. |
| CREATE_SE_TIMEOUT | SE creation timed out. |
| CREATED_SE | SE created. |
| CREATING_SE | Creating SE for virtual service. |
| DEL_NW_SE | Network(s) deleted. |
| DELETE_SE_FAIL | SE deletion failed. |
| DELETED_SE | SE deleted. |
| DELETING_SE | Deleting SE. |
| DISCOVERY_DATACENTER_DEL | vCenter Discovery Datacenter deleted from vCenter. |
| DUPLICATE_SUBNETS | IP subnet discovered in multiple networks. |
| ESX_HOST_POWERED_DOWN | vCenter Host powered down. |
| ESX_HOST_UNREACHABLE | Controller unable to communicate with hosts in vCenter. |
| GS_DOWN | GSLB service is down. |
| GS_UP | GSLB service is up. |
| GS_GROUP_DOWN | GSLB service group is down. |
| GS_GROUP_UP | GSLB service group is up. |
| GS_MEMBER_DOWN | GSLB service member is down due to GSLB health monitor failure. |
| GS_MEMBER_UP | GSLB service member is up due to GSLB health monitor success. |
| GSLB_DNS_STATUS | Provides visiblity such as new DNS assigned for GSLB functionality, existing DNS removed from GSLB functionality, no DNS assigned for GSLB. |
| GSLB_GS_STATUS | Provides visiblity such as a GSLB service member is down/up, there has been a transition in the overall status of the GSLB service. |
| GSLB_SITE_EXCEPTION_STATUS | A remote GSLB site is not reachable because of network connectivity or password issues. |
| GSLB_SITE_OPER_STATUS | If GSLB site is reachable, then = OPER_UP; if disabled, then = OPER_DISABLED, if unreachable, then = OPER_DOWN. |

| Event | Description |
|---|---|
| LICENSE_ADDITION_NOTIF | License addition. |
| LICENSE_EXPIRED | License has expired. |
| LICENSE_EXPIRY_NOTIF | License expiry notification. |
| LICENSE_REMOVAL_NOTIF | License removal. |
| LICENSE_USAGE_CORES | % of total licensed SE cores used. |
| LICENSE_USAGE_SERVERS | % of total licensed back-end servers used. |
| LICENSE_USAGE_THROUGHPUT | % of total licensed SE throughput used. |
| LICENSE_USAGE_VS | % of total licensed virtual services used. |
| MESOS_ACCESS_FAILURE | Mesos access failure. |
| MESOS_ACCESS_SUCCESS | Mesos access success. |
| MESOS_CREATE_SE_FAIL | SE create failed. |
| MESOS_CREATED_SE | Created SE. |
| MESOS_DELETE_SE_FAIL | SE delete failed. |
| MESOS_DELETED_SE | SE deleted. |
| MESOS_IMAGE_UPLOAD_FAILURE | Mesos image upload failure. |
| MESOS_IMAGE_UPLOAD_IN_PROGRESS | Mesos image upload in progress. |
| MESOS_IMAGE_UPLOAD_SUCCESS | Mesos image upload success. |
| MESOS_START_SE_FAIL | SE start failed. |
| MESOS_STARTED_SE | SE started. |
| MESOS_STOP_SE_FAIL | SE stop failed. |
| MESOS_STOPPED_SE | SE stopped. |
| MESOS_UPDATED_HOSTS | Mesos list of hosts has changed. |
| METRIC_THRESHOLD_UP_VIOLATION | Threshold exceeded for metric |
| METRICS_DB_DISK_FULL | Metrics DB Disk usage reached quota GB. |
| MGMT_NW_DEL | vCenter Management Network deleted from vCenter. |
| MGMT_NW_NAME_CHANGED | vCenter Existing New. |
| MODIFY_NW | Network(s) .. Modified for virtual service. |
| MODIFY_NW_FAIL | Modify network(s) failed on SE . Reason. |
| NEW_PROBABLE_SRVR | New Server matching filter in network for pool. |

| Event | Description |
|---|---|
| NO_HOST_AVAIL | No host available for placement of new SE virtual machine. |
| OPENSTACK_ACCESS_FAILURE | OpenStack access/setup failure. |
| OPENSTACK_ACCESS_SUCCESS | OpenStack access/setup succeeded. |
| OPENSTACK_IMAGE_UPLOAD_FAILURE | OpenStack image upload failure. |
| OPENSTACK_IMAGE_UPLOAD_SUCCESS | OpenStack image upload succeeded |
| OPENSTACK_IP_ATTACH_FAILURE | IP attach failure. |
| OPENSTACK_IP_ATTACHED | IP address attached to port. |
| OPENSTACK_IP_DETACH_FAILURE | IP detach failure. |
| OPENSTACK_IP_DETACHED | IP address detached from port. |
| OPENSTACK_LBPLUGIN_OP_FAILURE | OpenStack LBaaS plugin operation failure. |
| OPENSTACK_LBPLUGIN_OP_SUCCESS | OpenStack LBaaS plugin op success. |
| OPENSTACK_LBPROV_AUDIT_FAILURE | OpenStack LBaaS audit check failure. |
| OPENSTACK_LBPROV_AUDIT_SUCCESS | OpenStack LBaaS audit check success. |
| OPENSTACK_SE_CREATION_FAILURE | SE creation failure. |
| OPENSTACK_SE_DELETION_FAILURE | SE deletion failure. |
| OPENSTACK_SE_VM_CREATED | OpenStack SE created. |
| OPENSTACK_SE_VM_DELETED | OpenStack SE deleted. |
| OPENSTACK_SE_VM_DELETION_DETECTED | SE deleted in OpenStack. |
| OPENSTACK_SYNC_SERVICES_FAILURE | OpenStack sync services failure. |
| OPENSTACK_SYNC_SERVICES_SUCCESS | OpenStack sync services success. |
| OPENSTACK_TENANTS_DELETED | OpenStack tenants deleted. |
| OPENSTACK_VNIC_ADDED | New vNIC added to SE. |
| OPENSTACK_VNIC_ADDITION_FAILURE | vNIC addition failure. |
| OPENSTACK_VNIC_DELETION_FAILURE | vNIC deletion failure. |
| OPENSTACK_VNIC_REMOVED | vNIC removed from SE. |
| PKT_BUFF_ALLOC_FAIL | packet buffer allocation failure(s). |
| PKT_DROP_NO_PKT_BUFF | packet(s) dropped due to low packet buffer. |
| POOL_DOWN | Pool is DOWN. |

| Event | Description |
|---|---|
| POOL_HEALTH_CHANGE | Pool health score changed from, to. |
| POOL_SE_HA_ACTIVE | Pool is UP on SE. |
| POOL_SE_HA_COMPROMISED | Pool is DOWN on SE. |
| POOL_UP | Pool is UP. |
| REBALANCE_VS_MIGRATE | Action Migrate executed during Auto-Rebalance. |
| REBALANCE_VS_SCALEIN | Action Scale-In executed during Auto-Rebalance. |
| REBALANCE_VS_SCALEOUT | Action Scale-Out executed during Auto-Rebalance |
| REBOOT_SE | Rebooting unresponsive SE. |
| RM_DEL_NETWORK_FAIL | Deleting networks from SE failed. |
| SE_CONN_MEM_HIGH | Connection memory usage over Threshold % Current %. |
| SE_CPU_HIGH | CPU usage over Threshold % Current %. |
| SE_DISK_HIGH | Disk usage over Threshold % Current %. |
| SE_DOWN | SE marked down. |
| SE_DP_HB_FAILED | Dataplane heartbeat Failed() to SE. |
| SE_EV_GS_DOWN | GSLB service is DOWN. |
| SE_EV_GS_UP | GSLB service is UP. |
| SE_EV_GS_GROUP_DOWN | GSLB service group is DOWN. |
| SE_EV_GS_GROUP_UP | GSLB service group is UP. |
| SE_EV_GS_MEMBER_DOWN | GSLB service member is DOWN due to GSLB health monitor failure. |
| SE_EV_GS_MEMBER_UP | GSLB service member is UP. |
| SE_EV_POOL_DOWN | Pool is DOWN. |
| SE_EV_POOL_UP | Pool is UP. |
| SE_EV_SERVER_DOWN | Server is DOWN due to health monitor failure. |
| SE_EV_SERVER_UP | Server is UP. |
| SE_EV_VS_DOWN | Internal event generated by each SE on which the VS is placed to indicate that the Virtual Service is Down on that SE. |
| SE_EV_VS_UP | Internal event generated by each SE on which the VS is placed to indicate that the Virtual Service is Up on that SE. |
| SE_EXTERNAL_HM_RESTART | External health monitor process crashed. |

| Event | Description |
|---|---|
| SE_FATAL_ERROR | SE crashed. |
| SE_FLOW_TBL_HIGH | Flow table usage over Threshold % Current %. |
| SE_GROUP_CLUSTER_DEL | vCenter Cluster deleted from vCenter. |
| SE_GROUP_HOST_DEL | vCenter Host deleted from vCenter. |
| SE_GROUP_MGMT_NW_DEL | vCenter Management Network deleted from vCenter. |
| SE_HEALTH_CHANGE | SE health score changed from, to. |
| SE_HEALTH_CHECK_FAIL | SE failed health check. |
| SE_HEARTBEAT_FAILURE | SE Heartbeat failure. |
| SE_HM_EVENT_GHM_DOWN | GSLB health monitor instance for this GSLB service is down. |
| SE_HM_EVENT_GHM_UP | GSLB health monitor instance for this GSLB service is UP. |
| SE_HM_EVENT_SHM_DOWN | Health monitor instance for this server is DOWN. |
| SE_HM_EVENT_SHM_UP | Health monitor instance for this server is UP. |
| SE_MARKED_DOWN | SE unreachable - Marked down. |
| SE_MEM_HIGH | Memory usage over Threshold % Current %. |
| SE_PERSIST_TBL_HIGH | Persistence table usage over Threshold % Current %. |
| SE_PKT_BUFF_HIGH | Packet buffer usage over Threshold % Current %. |
| SE_POOL_DELETED | Pool deleted from SE. |
| SE_POWERED_DOWN | SE powered down by virtual infrastructure. |
| SE_REBOOTED | SE rebooted by user. |
| SE_SERVER_APP_CHANGED | The application (version) running in the Server (:) in Pool changed. |
| SE_SERVER_DELETED | Server (:) was deleted. |
| SE_SERVER_DISABLED | Server (:) in Pool was disabled |
| SE_SYN_CACHE_USAGE_HIGH | SYN cache usage over Threshold % Current %. |
| SE_SYN_TBL_HIGH | SYN table usage over Threshold % Current %. |
| SE_UP | SE connected to Controller. |
| SE_UPGRADING | SE upgrade in progress. |
| SE_VERSION_CHECK_FAILED | SE version does not match Controller version. |
| SE_VM_DELETED | SE VM deleted from virtual infrastructure. |

| Event | Description |
|---|---|
| SE_VM_PURGED | SE VM purged by user. |
| SE_VNIC_DHCP_IP_ALLOC_FAILURE | IP address allocation failure. SE vNIC Network MAC. |
| SE_VNIC_DUPLICATE_IP | Duplicate - IP vNIC IP, Remote MAC, Local MAC. |
| SE_VNIC_IP_ADDED | IP address added to vnic, Service Engine vNIC IP Mask Mode Namespace Network MAC. |
| SE_VNIC_IP_REMOVED | IP address removed from vnic, Service Engine vNIC IP Mask Mode Namespace Network MAC. |
| SERVER_AUTOSCALE_FAILED | Server Scale-Out failed. |
| SERVER_AUTOSCALE_IN | Scale-In of server(s) due to. |
| SERVER_AUTOSCALE_IN_COMPLETE | server(s) Scale-In complete. |
| SERVER_AUTOSCALE_OUT | Scale-Out of server(s) due to. |
| SERVER_AUTOSCALE_OUT_COMPLETE | server(s) launch complete. |
| SERVER_DELETED | Server(s) .. deleted from Virtual Infrastructure. |
| SERVER_DOWN | Server (:) in Pool is DOWN due to health monitor failure. |
| SERVER_DOWN_HA_COMPROMISED | Server (:) down due to health monitor failure by at least one but not all SEs. |
| SERVER_HEALTH_CHANGE | health score changed from to. |
| SERVER_UP | Server (:) in Pool is UP. |
| SERVER_UP_HA_ACTIVE | Server (:) is active. |
| SSL_CERT_EXPIRE | SSL certificate expires in days. |
| SSL_KEY_EXPORTED | SSL Private Keys are exported by. |
| SUMMARIZED_SUBNETS | Overlapping subnets detected in the cloud infrastructure. |
| SYSTEM_UPGRADE_ABORTED | system upgrade failed and was ended prematurely. |
| SYSTEM_UPGRADE_COMPLETE | system upgrade complete. |
| SYSTEM_UPGRADE_STARTED | system upgrade started. |
| UPGRADE_ALL_SE_DONE | Done upgrading SEs. |
| UPGRADE_ALL_SE_NOT_NEEDED | Upgrading SEs not needed. |
| UPGRADE_ALL_SE_START | Starting upgrade of SEs with virtual services. |
| UPGRADE_SE_DONE | SE upgrade done. |
| UPGRADE_SE_NOT_NEEDED | SE already upgraded. |
| UPGRADE_SE_START | Starting upgrade for SE. |

| Event | Description |
|---|---|
| UPGRADE_SE_VS_DISRUPTED | Virtual service upgrade disrupted. Reason. |
| UPGRADE_SE_VS_MIGRATE | Action Migrate executed during SE upgrade. |
| UPGRADE_SE_VS_SCALEIN | Action Scale-In executed during SE upgrade. |
| UPGRADE_SE_VS_SCALEOUT | Action Scale-Out executed during SE upgrade. |
| USER_LOGIN | User login from. |
| USER_LOGOUT | User logout from. |
| USER_PASSWORD_CHANGE_REQUEST | Password change request for email from IP. |
| VCA_ACCESS_FAILURE | VCA access/setup failure. |
| VCA_ACCESS_SUCCESS | VCA access/setup succeeded. |
| VCA_IMAGE_UPLOAD_FAILURE | VCA image upload failure. |
| VCA_IMAGE_UPLOAD_SUCCESS | VCA image upload succeeded. |
| VCENTER_ADDRESS_ERROR | Unable to contact vCenter. |
| VCENTER_BAD_CREDENTIALS | Bad user credentials when logging to vCenter. |
| VCENTER_VERSION_NOT_SUPPORTED | vCenter has a version which is not supported by the Controller. |
| VINFRA_DISC_CLUSTER | vCenter Num Clusters. |
| VINFRA_DISC_DC | vCenter Num Datacenter. |
| VINFRA_DISC_FAILURE | vCenter Discovery Failure in retrieving . Retrying. |
| VINFRA_DISC_HOST | vCenter Num Hosts. |
| VINFRA_DISC_NW | vCenter Num NWs. |
| VINFRA_DISC_VM | vCenter discovery found VMs. |
| VM_ADDED | A new virtual machine (non SE) was created. |
| VM_REMOVED | An existing virtual machine (non SE) was removed. |
| VS_ADD_SE | Virtual service added to SE. Role. |
| VS_ADD_SE_INT | Virtual service added to SE. Role. |
| VS_AWAITING_SE | SE not assigned to this virtual service even after seconds. |
| VS_CONN_LIMIT | connection(s) dropped due to virtual service connection limit. |
| VS_DOWN | Virtual service is DOWN. |
| VS_FSM_ACTIVE | Virtual service active. |

| Event | Description |
|---|---|
| VS_FSM_ACTIVE_AWAITING_SCALEOUT_READY | Virtual service awaiting Scale-Out ready status from SE. |
| VS_FSM_ACTIVE_AWAITING_SE_TRANSITION | Virtual service Active but awaiting more SEs. |
| VS_FSM_AWAITING_SE_ASSIGNMENT | Virtual service awaiting SE assignment. |
| VS_FSM_DISABLED | Virtual service Disabled. |
| VS_FSM_INACTIVE | Virtual service is inactive. |
| VS_FSM_PARTITIONED | Virtual service Partioned due to SE disconnect. |
| VS_FSM_PERMANENT_ERROR | Virtual Service Permanent Error. |
| VS_FSM_TRANSIENT_ERROR | Virtual service Transient Error. |
| VS_FSM_UNEXPECTED_EVENT | Virtual service FSM unexpected event. |
| VS_HEALTH_CHANGE | Virtual service health score changed from to. |
| VS_INITIAL_PLACEMENT_FAILED | Virtual service initial placement on a SE failed. |
| VS_MIGRATE_COMPLETE | Virtual service Migrate Done. |
| VS_MIGRATE_DONE | Virtual service Migrate Done. |
| VS_MIGRATE_FAILED | Virtual service Migrate Failed. |
| VS_MIGRATE_SCALEIN_DONE | Virtual service Scale-In Migrate Done. |
| VS_MIGRATE_SCALEIN_ERROR | Virtual service Scale-In Migrate Error. |
| VS_MIGRATE_SCALEOUT_DONE | Virtual service Migrate Scale-Out Complete. |
| VS_MIGRATE_SCALEOUT_ERROR | Virtual service Migrate Scale-Out Error. |
| VS_MIGRATE_STARTED | Virtual service Migrate Started. |
| VS_REMOVED_SE | Virtual service removed from SE. |
| VS_REMOVED_SE_INT | Virtual service removed from SE. |
| VS_RPC_FAILED_EVENT | Virtual service RPC Request failed. |
| VS_RPC_TO_RESMGR_FAILED_EVENT | Virtual service RPC to RM failed. |
| VS_RPC_TO_SE_FAILED_EVENT | Virtual service RPC to SE failed. |
| VS_SCALEIN_COMPLETE | Virtual service Scale-In Done. |
| VS_SCALEIN_DONE | Virtual service Scale-In Complete. |
| VS_SCALEIN_DONE_AWAITING_MORE_SE | Virtual service Scale-In Complete, but awaiting more Service Engines. |
| VS_SCALEIN_ERR | Virtual service Scale-In Error. |

| Event | Description |
|---|---|
| VS_SCALEIN_FAILED | Virtual service Scale-In Failed. |
| VS_SCALEOUT_COMPLETE | Virtual service Scale-Out Done. |
| VS_SCALEOUT_DONE | Virtual service Scale-Out complete. |
| VS_SCALEOUT_DONE_AWAITING_MORE_SE | Virtual service Scale-Out complete, but awaiting more Service Engines. |
| VS_SCALEOUT_ERR | Virtual service Scale-Out Error. |
| VS_SCALEOUT_FAILED | Virtual service Scale-Out Failed. |
| VS_SE_BOOTUP_FAIL | SE failed to boot. |
| VS_SE_HA_ACTIVE | Virtual service is UP on SE. |
| VS_SE_HA_COMPROMISED | Virtual service is DOWN on SE. |
| VS_SE_IP_FAIL | SE failed to acquire IP address on network(s) subnet(s). |
| VS_THROUGHPUT_LIMIT | packet(s) dropped due to virtual service bandwidth limit. |
| VS_UP | Virtual service is UP. |

# Alerts Notifications

The NSX Advanced Load Balancer actively monitors system health and tracks it through events or metrics. Over 500 data points for the system and virtual services are analyzed continuously by alert configurations to trigger alerts.

An important action for a newly generated alert is to notify an administrator of the issue. All notifications to external systems are sent from one or more Controllers and not from the Service Engines (SEs). Alerts are very useful in enabling system automation.

## Alerts Overview

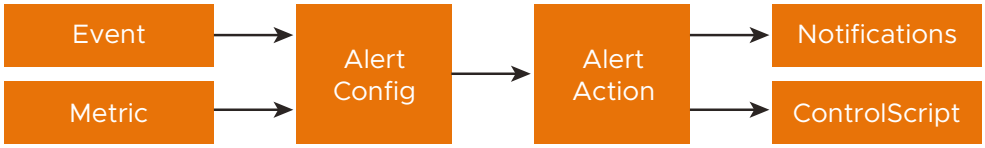
In its most generic form, alerts are a construct intended to inform administrators of significant events within the NSX Advanced Load Balancer system. Once an alert is triggered, it can be used to notify an administrator through one or more notification actions. Alerts can trigger ControlScripts to alter the system configuration or communicate to remote systems.

### Alert Scope

Alerts can be viewed in multiple places within the UI, including under **Virtual Services**, **Pools**, **Service Engine** and **All Alerts** pages. Navigate to **Operations > Alerts > All Alerts** to view all current alerts, logged across the system. Alert lists under SEs, Virtual Services, and Pools include alerts specific to the context.

Alerts are transient and can exist or be true only for a defined period. After the configured alert expiry time lapses, the alert is removed. The underlying event or metric that triggered the alert still exists as a permanent log of what has happened.

## Alert Workflow




Configuring custom alerts requires walking through the workflow to build the alert config (the alert's trigger) based on one or more metrics from a list of more than 500 events and metrics. When the alert configuration's conditions are met, the actions configured in the alert action are executed. The alert action can list the notifications to be generated and potentially call for further action, such as application autoscaling or execution of a ControlScript.

## Alert Config

The Alert Config page is used to define the triggers that will generate an alert. The NSX Advanced Load Balancer has a number of canned alert configs, which are used to generate alerts in a default deployment.

Events trigger alerts to actively highlight important information. Default alerts are created through the **Alert Config** page. These alerts configs can be modified or disabled, but not deleted. Alert configs are triggers that determine whether or not an alert should be generated.

To create a new alert config:

1 Navigate to **Operations > Alerts > Alert Config**.

2 Provide basic information such as **Name**, **Alert Configuration Status** and **Description.**

3 Choose the conditions that must be true to trigger an alert. The source can be an event or a metric. If multiple conditions exist, then all conditions must be true.

4 The following table lists details the input fields of the **Conditions** modal page:

| Field Name | Description |
|---|---|
| **Throttle Alert** | The alert must only be triggered once within the specified time frame. A value of 0 indicates there will be no time-based throttling. The timer begins once the alert is triggered. |
| **Source - Event** | An event triggers the alert. See the Events List for a list and brief description of all events. |
| **Object** | The type of object to listen for - a virtual service, SE, or pool.<br>Starting with version 20.1.3, all object types are available to be selected (through the CLI/API only). |

| Field Name | Description |
|---|---|
| **Instance** | Choose from a list of objects, based on the previously defined object type. |
| **Number of Occurrences** | The event must be seen X many times before the alert condition is met. |
| **Rolling Window** | When unchecked, an alert is triggered when the **Number of Occurrences** is met. If the **Number of Occurrences**value is set to 1, every occurrence will trigger the alert (the **Throttle Alert** can suppress any alerts beyond the first one). If the value for **Number of Occurrences** is set to a higher number, say x, the alert is triggered every x times the event happens.<br>When the **Rolling Window** is checked, a corresponding **Time Window** field must be populated. If the number of Occurrences is true within the specified window of time, the alert is triggered. Once the time window expires, the **Number of Occurrences** is reset and the counter begins again. |
| **Event Occurs** | Name of the event that would trigger the alert. |
| **Event Does Not Occur** | This field is optional. When set, the selected event must not be true during the same time window as the event defined in the **Event Occurs** field. |
| **Source - Metric** | When chosen, Metrics trigger the alert. When the Source is set to Metric, a few different options are presented:<br>**Metric Occurs**: Select the desired metric.<br>**Comparator**: The metric entry is compared through the **greater than or equals**, **equals**, or **less than or equals** to the averaged entry in the **Value** field over the specified number of seconds.<br>**Value**: Enter the scalar portion of the metric. For example, to specify “2 milliseconds,” “2” is the scalar and “milliseconds” is the unit of measure. After a selection is made in the Metric Occurs field, the controller auto-populates the unit portion of the Value field. The units of measurement for the various metrics are documented in the Metrics Query API.<br>**Duration**: Enter a value greater than 5 minutes. The period should be in order of 10s of minutes.<br>**Add New Metric Rule**: Additional rules can be specified. When multiple rules exist, all of them must be true. |

1 Enter values for the following fields in the Action modal window:

   a **Alert Action**: Alert action defines the type of notifications to generate or other tasks resulting from the triggered alert.

   b **Alert Expiry Time**: The triggered alert will be visible in the web interface for this duration of time, after which it is deemed expired and deleted.

## Notifications

Alerts can be sent to four notification destinations. The first is the local UI. These notifications show up as colored bell icons in the Controller UI to indicate that an alert has occurred. The other three notifications namely, email, syslog, and SNMP v2c traps can be configured in the **Operations > Notifications** page.

Notifications can be classified into Local notifications, SNMP traps, Syslog and Email.

For more information on notification types, see Types of Notifications

## ControlScript

ControlScripts are Python-based scripts, executed from the Controller. They can be used to alter the NSX Advanced Load Balancer configuration or communicate to remote systems (such as email, syslog, and/or SNMP servers). For example, if it is detected that an IP address is sending a SYN flood attack, the alert action can notify administrators by email and also invoke a ControlScript that adds the offending client to a blocklist IP group, attached to a network security policy, for rate shaping or blocking attackers.

## Alert Actions

When an alert is triggered, alert configurations call the corresponding **alert actions**. The reusable **alert config** object defines what should happen for the triggered alert.

### Alert Actions

Alerts actions can be used to define the alert level, and notify administrators through the NSX Advanced Load Balancer UI(default mode), email, Syslog or SNMP traps. The alert action can trigger a **ControlScript** or an autoscale policy to scale the application in or out, to a particular level (SEs in the SE group, members of a server pool).

They can also be used to effect automation through:

- Application autoscaling (e.g., SE scale-out or sacle-in, server pool autoscaling).
- Execution of a ControlScript.

An alert action can specify any combination of these actions.

To create a new alert action navigate to **Operations > Alerts > Alert Actions**. The following table explains the list of fields to be entered in the **New Alert Action** screen:

| Field Name | Description |
|---|---|
| **Name** | User-friendly name. |
| **Only Generate External Alerts** | By default, the controller appends an entry in the alert log, which is visible to administrators in the UI. Checking the **Only Generate External Alerts** box disables this logging. Alerts can still be sent externally through any combination of the four methods listed (email, syslog, SNMP, ControlScript). |

| Field Name | Description |
|---|---|
| **Autoscale Trigger** | Checking this box engages the Autoscale Manager. |
| **Alert Level** | High, medium, or low. Provides a way of classifying the alert to the remote system. For local notifications within the Controller UI, the alerts show as a different color to denote their severity. |
| **Email** | Send the alert as an email by selecting a previously created Email Notification. |
| **Syslog** | Send the alert to a syslog server (or servers) by selecting a previously defined Syslog Notification. |
| **SNMP Trap** | Send the alert as a trap an SNMP server (or servers) by selecting a previously defined SNMP Trap Notification. |
| **ControlScript** | Launch a custom ControlScript, which is a Python script to be executed on the Controller. |

### Syslog Audit Persistence

To stream alerts of events for audit compliance, starting with NSX Advanced Load Balancerversion 20.1.3, a new alert action, the **Syslog-Audit-Persistence** is created for streaming events to external **rsyslog** servers.

Use **Syslog-Audit-Persistence** as a template, and configure the alert action as required.

To edit **Syslog-Audit-Persistence**:

1 Navigate to **Operations > Alerts > Alert Actions**. Click the edit icon.

2 In the **Edit Alert Action** screen, update general information like **Alert Level** and **Email**.

3 Select the **Syslog** notification configuration to use when sending alerts through Syslog or click **Create Syslog Notification**.

4 In the **New Syslog Notification** screen, update the **Name**, **Syslog Server** and Port.

5 Click **Save**.

### Syslog Messages for TCP

When configured from the UI, syslog message streaming defaults to using UDP. To use TCP, the configuration has to be changed from the CLI as follows:

```
[*:alert-ctlr]: alertsyslogconfig> syslog_servers index 1
[*:alert-ctlr]: alertsyslogconfig:syslog_servers> no udp
+--------------------+---------------+
| Field              | Value         |
+--------------------+---------------+
| syslog_server      | 10.10.0.235   |
| syslog_server_port | 514           |
| udp                | False         |
|format              | SYSLOG_LEGACY |
| tls_enable         | False         |
```

```
| anon_auth          | False         |
+--------------------+---------------+
[*:alert-ctlr]: alertsyslogconfig:syslog_servers> save
[*:alert-ctlr]: alertsyslogconfig> save
+----------------------+------------------------------------------------------------+
| Field                | Value                                                      |
+----------------------+------------------------------------------------------------+
| uuid                 | alertsyslogconfig-c39ad76c-4630-4c87-8c56-0d6df5ffc78f |
| name                 | Pybot-Syslog-Cfg                                           |
| syslog_servers[1]    |                                                            |
|   syslog_server      | 10.10.0.235                                                |
|   syslog_server_port | 514                                                        |
|   udp                | False                                                      |
|   format             | SYSLOG_LEGACY                                              |
|   tls_enable         | False                                                      |
|   anon_auth          | False                                                      |
| tenant_ref           | admin                                                      |
+----------------------+------------------------------------------------------------+
[*:alert-ctlr]: >
```

### Using Syslog Audit Persistence

The updated **Syslog-Audit-Persistence** can be used when configuring an alert as shown below:

1 From the Controller UI, navigate to **Operations > Alerts > Alert Config**.

2 Configure the **Basics** and the **Conditions** sections as required.

3 Under **Actions**, select **Syslog-Audit-Persistence** as the **Alert Action**.

4 Click **Save**.

The alert config will trigger **Syslog-Audit-Persistence** which sends notifications and executes a ControlScript.

## Types of Notifications

Notifications communicated by the Controller include Local Notifications, SNMP Traps, Syslog, and email.

### Local Notifications

Local notifications are marked by alert action with an alert priority for categorizing the alerts. Though high severity alerts are more important than medium or low alerts, there is no functional difference between them. The color in the GUI reflects only the severity. The **Alert Actions** object determines which types of notifications are generated for a new alert. Though this object has an option to disable notifications logged to the internal database through the **Only Generate External Alerts** check box, it does not explicitly call out these notifications.

### SNMP Traps

Alerts can be sent through SNMP traps using SNMP v2c. Multiple trap servers can be defined.

Configuring SNMP traps is solely for sending alerts to an SNMP trap server and not for polling the NSX Advanced Load Balancer SNMP OIDs.

Though traps are sent from the Controller Cluster leader, the leadership role can move to either follower Controller after a failure. So the external SNMP server must be configured to allow traffic from any one of the three Controllers in the cluster, i.e., all three addresses should be in the SNMP server's allowed-access list. The firewall rules must be configured to allow UDP traffic destined to port 162 on the SNMP trap server from any of the three cluster member's IP addresses.

## Syslog

Syslog messages can be sent to one or more syslog servers. Communication is non-encrypted through UDP, using a customizable port. According to RFC 5426, syslog receivers accept syslog datagrams on port 514 (default) and can also be configured to listen on a different port. Configuring **syslog notifications** pushes alerts to syslog servers. It does not export the virtual service logs. The logs can be pulled from an external logging system through the API, or scripted to be pushed from the Controller to a remote log system.

## Email

Alert Actions can be configured for sending alerts to administrators through email. These emails are sent directly to administrators or reporting systems that accept email. The Controller must have a valid DNS and a default gateway or route configured, to resolve the destination mail server and properly forward the messages. Information regarding the SMTP server and the sender must be configured in the **Administration > Settings > Email/SMTP** page before email notifications can be sent.

# Application Monitoring

3

This section lists the application monitoring analytics that delivers performance insights to quickly troubleshoot any root cause issues in the NSX Advanced Load Balancer.

This chapter includes the following topics:

- Virtual Service Analytics
- Pool Analytics
- Service Engine Analytics
- Virtual Service Application Logs

## Virtual Service Analytics

This section covers the analytics that help you assess the virtual service performance metrics. The charts and metrics reflect the selected display time.

The virtual service analytics comprises the following sections:

- End-to-End Timing
- Metrics Tiles
- Chart Pane
- Overlays Pane
  - Anomalies
  - Alerts
  - Config Events
  - System Events

### End-to-End Timing

The end-to-end timing provides an overview of the quality of the end-user experience and the occurrence of any slowdowns. The chart breaks down the time required to complete a single transaction, such as an HTTP request. It breaks total transaction time into the following components:

| Field | Description |
|---|---|
| **Client RTT** | Average network TCP latency between the client and NSX Advanced Load Balancer for all clients, both local (within the same datacenter) and remote (Internet). This metric indicates the duration it takes to establish connections and return acknowledgments. This number will often be higher than an ICMP ping. |
| **Server RTT** | Round-trip latency for SE-to-server traffic. An abnormally high server RTT indicates that the network is saturated or that a TCP stack cannot quickly establish new connections due to heavy load. |
| **App Response** | The time taken for the server to respond. It includes the time the server took to generate content, potentially fetch backend database queries, and begin transferring the response back to the NSX Advanced Load Balancer. This metric is only available for a L7 virtual service. |
| **Data Transfer** | Average time required for the server to transmit the requested file. It might vary greatly depending on the size of objects requested and the latency of the server network. The larger the file, the more TCP round trip times are required due to acknowledgments (ACKs), directly impacted by the client and server RTT. This metric is only used for a L7 virtual service. |
| **Total Time** | Request-response time.<br>This is the most critical end-to-end timing to watch, because it is the sum of the other four metrics. If the **Total Time** is consistently low, the application is probably serving traffic successfully. |

## Metrics Tiles

Metrics tiles continuously display time-averaged data. Clicking a particular metric tile will display the corresponding data over time in the chart. The following are the available metrics tiles:

| Tile Name | Description |
|---|---|
| **End-to-End Timing** | Charts the end-to-end timing by default. It governs what appears in the chart display. It displays a color-coded plotted timing to indicate the client RTT, server RTT, data transfer, and app response times at any recorded instant. |
| **Throughput** | Total bandwidth passing through the virtual service in Mbps. Pointing over this graph displays the throughput in Mbps for the selected time. **Throughput** is measured as bytes transferred between the client and SE, not including data transferred between SE and the servers. |
| **Open Connections** | The number of TCP client connections or UDP transactions currently in an open state. UDP transactions are counted even though they are technically connection-less. The number of open connections does not necessarily translate into the number of unique clients, as protocols such as HTTP 1.1 typically open six connections per client browser. If the **Maximum Concurrent Connections** setting has been set for a virtual service, a horizontal red bar superimposed over the chart pane will show the maximum number of connections allowed. For more information, see Rate Shaping and Throttling Options. |
| **Conns** | The average rate of connections completed per second. The tile also summarizes the percentage of connections resulting in errors. Pointing over the tile shows the rates of good, lossy, and bad connections.<br>■ Lossy connections experience issues such as retransmissions, zero window size events, or high connection setup times.<br>■ Bad connections are those which terminate ungracefully. Connection closed abnormally.<br>■ Requests ended abnormally will result in bad connections. |

| Tile Name | Description |
|---|---|
| **Requests** | Number of responses to requests per second. It breaks down successful requests versus errors (such as 4xx or 5xx errors).<br>Within the chart pane, you can toggle radio buttons to break down the errors by whether they are generated by the server or by the NSX Advanced Load Balancer. For instance, if no servers are available in the pool, NSX Advanced Load Balancer will send out an HTTP 500 status code. The **Client Logs** section will provide more details on specific errors. This metric is only available for HTTP virtual services. |
| **HTTP vs HTTP2 Requests** | Displayed as requests-per-second rates. |

For more information on the chart and overlays pane, see Chart Pane and Overlays Pane.

# Pool Analytics

The pool analytics displays the pool performance metrics filtered by a selected period.

The pool analytics comprise of the following sections:

- End-to-End Timing
- Metrics Tiles
- Chart Pane
- Overlay Pane
  - Anomalies
  - Alerts
  - Config Events
  - System Events

## Pool End-to-End Timing

This section provides a high-level overview of the quality of the end-user experience and the occurrence of any slowdowns. The chart breaks down the time required to complete a single transaction, such as an HTTP request.

The End-to-End pane displays the following components:

| Name | Description |
|---|---|
| **Server RTT** | This is SE to server round trip latency. An abnormally high server RTT indicates either that the network is saturated or that a server's TCP stack is overwhelmed and cannot establish new connections quickly. |
| **App Response** | The time the servers take to respond. This includes the time the server took to generate content, potentially fetch back end database queries or remote calls to other applications, and begin transferring the response back to NSX Advanced Load Balancer. This time is calculated by subtracting the server RTT from the time of the first byte of a response from the server. If the application consists of multiple tiers (such as web, applications, and database), then the **App Response** represents the combined time before the server in the pool began responding. This metric is only available for a Layer 7 virtual service. |

| Name | Description |
|---|---|
| **Data Transfer** | Represents the average time required for the server to transmit the requested file. This is calculated by measuring from the time the SE received the first byte of the server response until the client has received the last byte, which is measured as when the last byte was sent from the SE plus one half of a client round trip time. This number varies depending on the size of objects requested and the latency of the server network. The larger the file, the more TCP round trip times are required due to ACKs directly impacted by the client RTT and server RTT. This metric is only used for a Layer 7 virtual service. |
| **Total Time** | The time it takes for a client to receive a response after sending a request. It is the most critical end-to-end timing number to watch because it is the sum of the other four metrics. If it is consistently low, the application is successfully serving traffic. |

For more information on the chart and overlays pane, see Chart Pane and Overlays Pane.

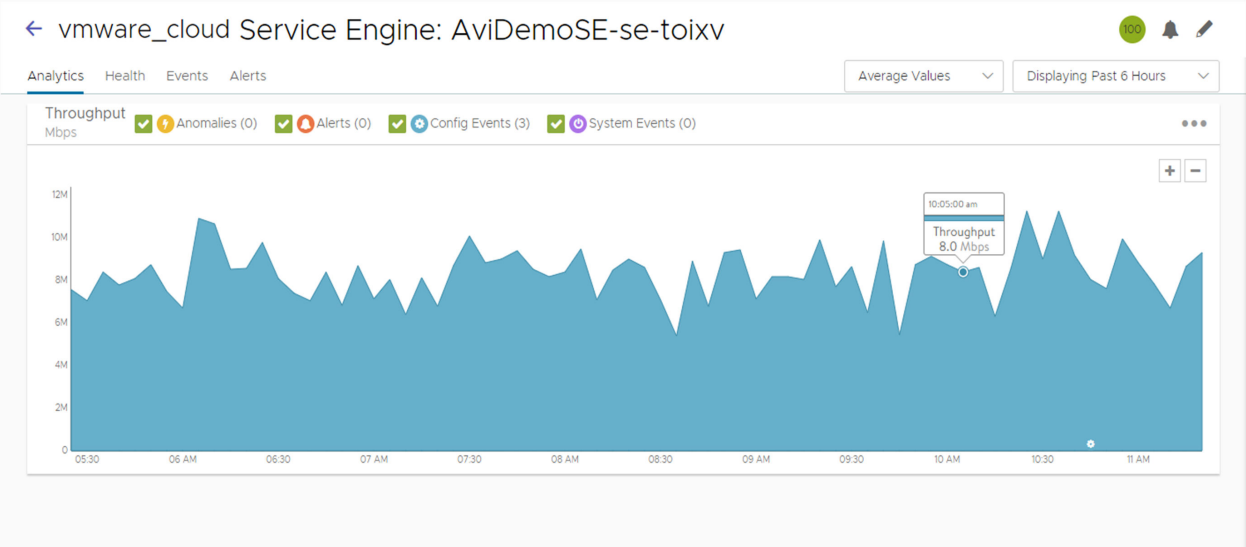
# Service Engine Analytics

The SE analytics provides the insights into various performance metrics over a designated period.

The SE analytics comprises of the following sections:

For more information on metrics, see SE Metrics.

## Chart Pane



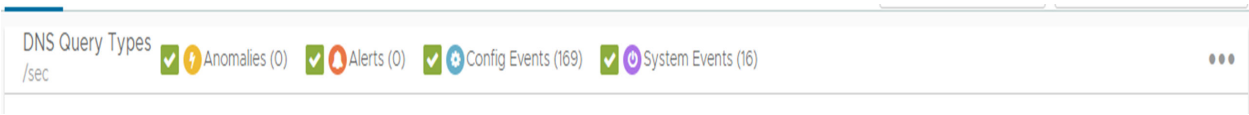

The chart pane displays a detailed historical chart of the selected **Metric Tile** for the current virtual service, pool, or SE. Point anywhere in the chart to display the results for that selected time in a pop-up window. Click within the chart to freeze the pop-up at that point in time. This helps to anchor a data point when the chart scrolls leftward with updates over time.

The radio buttons in the top right are used to customize the data that need to be excluded from the chart.

## Overlays Pane

The overlays pane allows you to overlay icons signifying important events within the timeline of the chart pane. This feature helps you correlate anomalies, alerts, configuration changes, and system events according to the changes in traffic patterns.

Clicking an **Overlay** button toggles the overlay items in the chart pane. Each overlay item lists the number of instances of that event type within the selected period. Clicking each overlay item will display additional data below the **Overlay Item** bar.

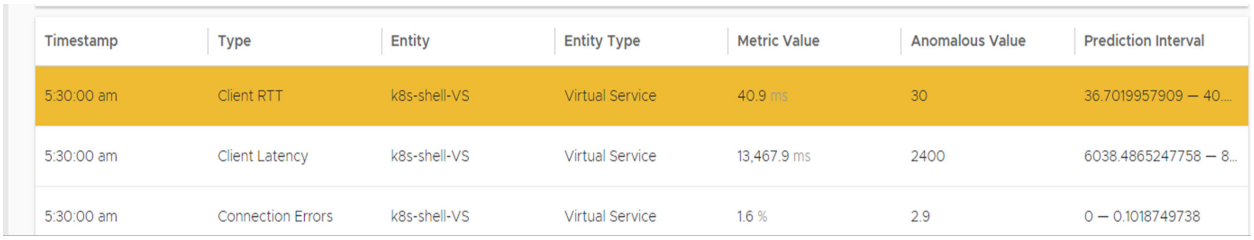



The items available in the overlay pane are:

| Item Name | Description |
|---|---|
| **Anomalies** | Display anomalous traffic events, such as a spike in server response time, with corresponding metrics collected during that period. |
| **Alerts** | Displays filtered system-level events that are critical enough to notify an administrator. |
| **Config Events** | Display configuration events, which track configuration changes made to the NSX Advanced Load Balancer by an administrator or an automated process. |
| **System Events** | Display system events, which are raw data points or metrics of interest.<br>System events can be noisy, and are best used by alerts which filter and classify these raw events by severity. |

## Anomalies Overlay

The **Anomalies Overlay** displays the periods of abnormal traffic detection based on its moving averages. Changing the time interval will provide greater granularity and potentially show more anomalies. Selecting the **Anomalies Overlay** check box displays yellow icons in the chart pane. Clicking on a yellow icon will display additional information related to that anomaly as shown below:



| Timestamp | Type | Entity | Entity Type | Metric Value | Anomalous Value | Prediction Interval |
|---|---|---|---|---|---|---|
| 5:30:00 am | Client RTT | k8s-shell-VS | Virtual Service | 40.9 ms | 30 | 36.7019957909 — 40.... |
| 5:30:00 am | Client Latency | k8s-shell-VS | Virtual Service | 13,467.9 ms | 2400 | 6038.4865247758 — 8... |
| 5:30:00 am | Connection Errors | k8s-shell-VS | Virtual Service | 1.6 % | 2.9 | 0 — 0.1018749738 |

| Name | Description |
|---|---|
| **Timestamp** | Date and time when the anomaly was detected. It might either span the entire duration of the anomaly or merely be near the same time window. |
| **Type** | The specific metric deviating from the norm during the anomaly period.<br>To be included, the metric deviation must be greater than 4 sigma. Numerous metrics, such as CPU utilization, bandwidth, or disk I/O might trigger anomalous events. |

| Name | Description |
|---|---|
| **Entity** | Name of the specific object that is reporting this metric. |
| **Entity Type** | Type of entity that caused the anomaly. This might be one of the following:<br>■ Virtual Machine (server): These metrics require NSX Advanced Load Balancer to be configured for either read or write access to the virtualization orchestrator such as vCenter or OpenStack.<br>■ Virtual Service<br>■ SE |
| **Metric Value** | The value of the metric. |
| **Anomalous Value** | Value of the anomaly. |
| **Prediction Interval** | Acceptable range for the metric. |

During times of abnormal traffic, the NSX Advanced Load Balancer records any deviations and provides insights into the root cause of the anomaly.

**Note** Anomalies are not recorded if the statistics interval is set to `Real Time` in the **Displaying** drop-down menu.

## Alerts Overlay

**Alerts Overlay** displays the results of any events that meet the filtering criteria defined by **Operations > Alerts**. It notifies the administrators about critical information or changes to a site that requires immediate attention.

Alerts are transitory, as they will expire after a defined time. For instance,NSX Advanced Load Balancer might generate an alert if a server is down and then allow that alert to expire after a specified duration once the server comes back online. The original event remains available for troubleshooting purposes later. Clicking the **Alerts** icon in the **Overlay Items** bar displays any red alerts in the chart pane. Selecting one of these chart alerts will display the below information under the **Overlay Items** bar:

| Name | Description |
|---|---|
| **Timestamp** | Indicates when the alert occurred. |
| **Resource Name** | Name of the object that is reporting the alert. |
| **SeverityLevel** | Use the priority level to determine whether additional notifications should occur, such as sending an email to administrators or sending a log to Syslog servers. The three severity levels are:<br>■ High: red<br>■ Medium: yellow<br>■ Low: blue |
| **Summary** | Briefly describes the event. |

| Name | Description |
|---|---|
| **Actions** | Dismiss the alert to remove it from the list and also remove the corresponding alert icon from the chart pane. Dismissing an alert here is the same as dismissing it through the bell icon at the top of the screen next to the health score or dismissing it using the **Alerts** tab. |
| **Edit** | Opens the alert filter configuration, which can be modified to make NSX Advanced Load Balancer more or less sensitive to generating new alerts. |
| **Expand/Contract** | Clicking the plus **(+)** or minus sign **(-)** for an alert either expands or contracts a sub-table showing more detail about the alert. This will typically show the original events that triggered the alert. |

## Configuration Events Overlay

The **Config Events** overlay displays configuration events, like changing the NSX Advanced Load Balancer configuration by adding, deleting, or modifying a pool, virtual service, or SE, or an object related to the object being inspected. If traffic dropped off at precisely 10:00 a.m., and at that time an administrator made a change to the virtual services security settings, the cause of the change in traffic was due to the (mis)configuration.

Clicking the **Config Events** icon in the **Overlay Items** bar displays any blue **Config Event** icons in the chart pane. Selecting one of these icons will display the following information below the **Overlay Items** bar:

| Field Name | Description |
|---|---|
| **Timestamp** | Date and time when the configuration change occurred. |
| **Event Type** | Always be scoped to configuration event types. |
| **Resource Name** | Name of the object that has been modified. |
| **Event Code** | There are three event codes:<br>■ CONFIG_CREATE<br>■ CONFIG_UPDATE<br>■ CONFIG_DELETE |
| **Description** | Brief description of the event. |
| **+/-** (Expand/Contract) | Expands or contracts a sub-table showing additional detail about the event.<br>When expanded, It shows a color-coded comparison of the previous configuration and the new configuration, as follows:<br>■ Additions to the configuration, such as adding a health monitor, will be highlighted in green in the new configuration.<br>■ Removing a setting will be highlighted in red in the previous configuration.<br>■ Changing an existing setting will be highlighted in yellow in both the previous and new configurations. |

## System Events Overlay

This overlay displays the system events relevant to the current object, such as a server changing status from `UP` to `DOWN` or the health score of a virtual service changing from `50` to `100`.

Clicking the **System Events** icon in the **Overlay Items** bar displays **System Event** icons in the chart pane. Clicking any **System Event** icon in the chart pane displays a detailed table of information with the below columns:

| Field Name | Description |
|---|---|
| **Timestamp** | Date and time when the system even occurred. |
| **Resource Name** | Name of the object that triggered the event. |
| **Resource Type** | Type of resource involved, e.g., virtual service, pool, pool server, etc. |
| **Event Code** | High-level definition of the event, such as `SERVER_HEALTH_DEGRADED`, `POOL_HEALTH_DEGRADED`, or `VS_UP`. |
| **Description** | Brief of the system event. |
| **+/-** | Expands or contracts the detailed information of an event. |

# Virtual Service Application Logs

This section explains the steps to enable, configure and analyse the virtual service application logs.

Virtual services and pools log client-to-application interactions for TCP connections and HTTP requests/responses. These logs can be indexed, viewed, and filtered locally within the NSX Advanced Load Balancer. It is helpful in troubleshooting and surfacing insights on end-user experience and success of the application.

## Enabling Logs

See the **Analytics** tab of the **Create Virtual Service** pop-up window for configuring, enabling, filtering, and/or disabling client logs.

## Significant Logs

NSX Advanced Load Balancer automatically logs common network and application errors under the umbrella of significant logs. These significant logs can also include entries for lesser issues, such as transactions that completed successfully but took a considerably longer time.

These logs include:

- HTTP errors, such as server or NSX Advanced Load Balancer-originated 4xx and 5xx errors Network errors, such as aborted connections, abnormal latency, or out of order packets.
- See Log Events for a list of error events that trigger a significant log.

You can omit the errors from the logs list by editing the analytics profile used by the virtual service.

## Full Client Logs

In addition to significant logs, you can configure a virtual service to log all client connections or HTTP requests. The **Full Client Logs** option includes any significant logs, custom full log filters, and logs generated by custom policies or DataScripts.

By default, a new virtual service is configured to provide full client logs for the first 30 minutes, then drops down to a reduced logging level by capturing significant logs only.

From the **Analytics** tab, **Full Client Logs** can be enabled for the virtual service, either temporarily or permanently.

You can also specify full client log filters for IP addresses or URIs that are recommended when capturing essential information from busy production systems. An additional logging level is provided by enabling the **All Headers** option in a client log filter. This option will capture all headers from the client and server within the logs. This will significantly impact the log size, as some applications send as much as 30k within a single header.

The **All Headers** option is beneficial for quick troubleshooting to see what each side of the connection is sending and receiving. The NSX Advanced Load Balancer pulls logs from the SEs and indexes them on the Controllers only when an administrator attempts to view full client logs for the virtual service or pool. This takes anywhere from a few seconds to hours to process. Logs will be viewable while the indexing process is performed in the background. This time depends on network latency from the SEs to the Controllers, the volume of logs, and the hardware used by the Controller for performing the resource-intensive task of indexing the data.

## Rotation Out of Unrequested Logs

Capturing all logs can consume significant resources. Hence, unrequested logs are rotated out of the SE's storage from time to time. The allocation for raw log storage on an SE is variable, but a minimally configured SE with two vCPUs, 2 GB memory, and 10 GB storage can store about 8 million logs. Adding more resources (CPU, memory, and especially disk space) to the Controllers and SEs will extend the volume of logs per second and the duration to store the logs.

## Logging During Heavy Load

In a very busy system with high volumes of requests per second, NSX Advanced Load Balancer might temporarily degrade to capturing only a sampling of requests. If the system is set up with redundant Controllers, the task of indexing logs for multiple virtual services is automatically shared across the Controllers to better utilize resources.

## Client User IDs

User IDs for a virtual service are incorporated into UI displays when an auth profile is attached to the virtual service. To achieve this:

1 Enable Basic HTTP Authentication in the virtual service config (**Edit VS > Advanced**).

2 Configure and attach an auth profile to the virtual service.

For further information, see:

- HTTP Basic Authentication section of Create a Virtual Service
- Auth Profile
- HTTP Authentication Settings section of the LDAP Authentication

## Log Navigation

This section explains the logs and sections of log entries.

To access the client logs for a virtual service:

- Navigate to **Applications > Virtual Services**.
- Click a virtual service name.
- Click **Logs**.

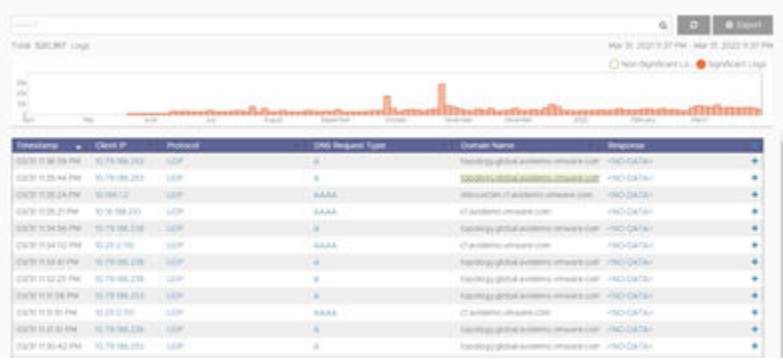
The page displays the following fields:

| Name | Description |
|---|---|
| **Search** | Allows you to filter the log entries using free-form text, keywords or a formal search syntax. |
| **Export** | Downloads upto 10,000 logs in CSV format. |
| **Refresh** | Updates the logs for the selected time. |
| **Number and Range** | Total number of log items being displayed. The date/time range of those logs appear beneath the **Search** field. The number will reflect how many logs passed the filter. |
| **Clear Selected** | If one or more filters have been added to the **Search** field, click the gray**X** icon near the search bar to remove those filters. To clear individual filters, click the white on blue **X** for that filter. |
| **Histogram** | Shows the number of client log entries over the selected time.<br>■ Green bars shows **All Logs**, or successful transactions.<br>■ Red bars shows **Significant Logs** or errors.<br>■ Orange question marks above the bars indicates the **Missing Logs**. |

**Note**

- Click and drag inside the histogram to refine the date/time which further filters the logs.
- Clicking the magnifying glass icon in the drawn box will zoom the Histogram to that view. This expands the drilled in time to zoom in to the width of the Histogram, and also changes the **Displaying Time** drop-down menu to **Custom**. To return to the previously selected period, use the **Displaying** drop-down menu.

The primary table on the **Logs** tab displays the list of individual log entries as shown below:



| Field Name | Description |
|---|---|
| **Timestamp** | Date and time of the event. |
| **Client IP** | The source IP address of the client to which the log entry applies. |
| **Protocol** | Type of protocol used such as TCP, UDP etc. |
| **DNS Request Type** | Type of HTTP method for the request, such as GET, POST, or HEAD. |
| **Domain Name** | Name of the network domain. |
| **Response** | For HTTP, the response code will be of type 1xx, 2xx, 3xx, 4xx, or 5xx. This is usually a reflection of the response sent by the server, but it may instead be a response created by the NSX Advanced Load Balancer. Expand the log to determine whether the server or the NSX Advanced Load Balancer generated the response. |
| **+/-** icon | Expands or collapses a view into logs. |

## Log Analytics

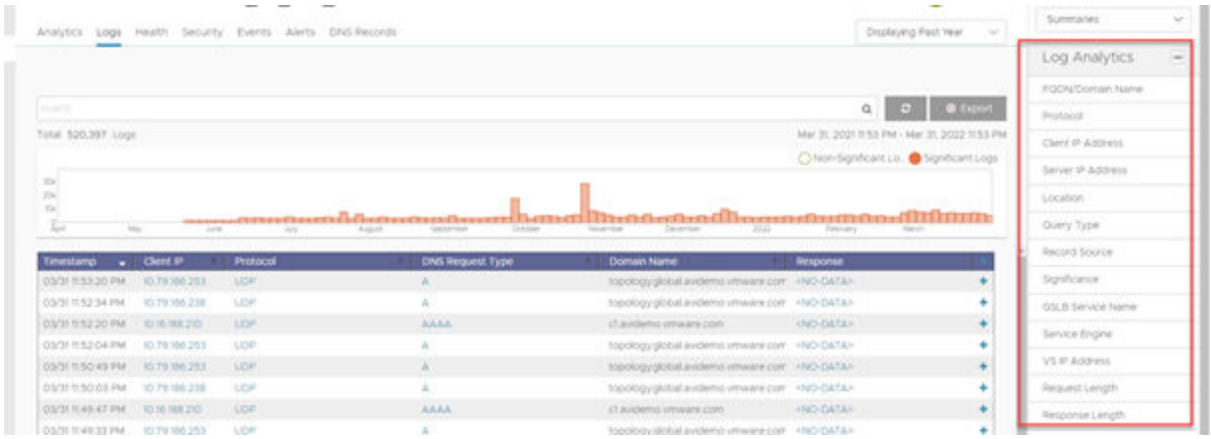
This section describes the log analytics and its tiles.

The **Log Analytics** area displays a series of prebuilt filters that summarize the client logs in a pop-up window according to the selected summary filter. For example, clicking the **IP Address summary** tile will display the most active client IP addresses that have requested the virtual service within the log display period.

The **Log Summaries** reflect the currently applied filters, including the displayed log period and the Non-Significant/Significant setting.

Client information is learned from IP addresses or the client's `HTTP User-Agent` string. Adding a search filter such as `client_location="US"` will filter the summaries to reflect only the results for clients coming from an IP address within the United States.

### Log Analytics Tiles




## Expanded Logs

This section describes the fields in the expanded log table.

Clicking the **+** plus icon on the right of the logs table expands an individual log. This provides an in-depth view of the specific connection log or the HTTP request and response log.

The following are displayed in the expanded table:

| Field | Description |
|---|---|
| **End to End Timing** | The bar is similar to the **Analytics** tab of the virtual service Details page, though it also contains arrows indicating the HTTP response code. This data is specific to this single connection or HTTP request, whereas the Analytics tab for the virtual service shows an aggregate of all connections or requests. If the arrow under Server RTT is a 0, then no response was received from the server. This could be due to an error such as a timed out server response, or because the request was served by the NSX Advanced Load Balancer, such as caching or a policy. |
| **IP Addresses** | Under End-to-End Timing, IP addresses and service ports indicate the client source address and port being used to initiate a transaction to the virtual service IP address and destination service port. The second address under the (LB) icon is the NSX Advanced Load Balancer source NAT (SNAT) address and source port that is used for communicating with the destination server's pool, name, IP address and port. |

| Field | Description |
| --- | --- |
| **Client data** | The column under the **Client** icon displays the following client information:<br>■ **Client IP**: The source IP address and service port of the client.<br>■ **Location**: The country of origin for the IP address or Internal for private IP addresses. This field can also show custom IP group names.<br>■ **Operating System**: The OS of the connecting device. HTTP only.<br>■ **Device**: The type of connecting device, such as computer, table, or phone. HTTP only.<br>■ **Browser**: The web browser of the connecting device. HTTP only.<br>■ **SSL Version**: The negotiated version, such as SSL 3.0, TLS 1.0, TLS 1.1, or TLS 1.2. SSL terminated HTTP traffic only.<br>■ **Certificate Type**: RSA or Elliptic Curve (EC) certificate used for the connection. SSL terminated HTTP traffic only.<br>■ **Perfect Forward Secrecy**: Did the client negotiate a cipher which protects the connection from later decryption through hijacked keys. SSL terminated HTTP traffic only.<br>■ **Start Time**: The time the connection was established or the request was received. |

| Field | Description |
| --- | --- |
| **LB data** | The following information appears under the **LB** icon in the middle column:<br>■ **Virtual Service IP**: The listening virtual service's IP and port.<br>■ **Server Conn IP**: The source IP address and port used as the source NAT address on the server side of the connection.<br>■ **End time**: When the log was generated. It normally occurs when the request or connection was completed. The logs are generated for currently active, long-lived connections. Logs generated during an open connection will be updated periodically or when the connection closes.<br>■ **Service Engine**: The SE and corresponding vCPU that was used to process the request or connection.<br>■ **Persistent Session ID**: Persistent Session ID for the request will be displayed even if persistence is not enabled.<br>■ **Response Length**: The size of the response, such as HTTP payload plus headers returned by the NSX Advanced Load Balancer to the client. This size may be different from the server Response Length in the server column due to SSL padding, Javascript insertion (when **Client Insights** is set to `Active`), compression, TCP maximum segment size differences, or a number of other features.<br>The following fields appear only if applicable:<br>■ **Cache Hit**: This is true if the HTTP request was served by the NSX Advanced Load Balancer cache. This field will not be shown if caching is disabled.<br>■ **Compression**: If the NSX Advanced Load Balancer compressed the response content, this will show the percent by which the content was able to be compressed.<br>■ **Policy Rule**: If a policy has been applied to the virtual service, any rules that were executed will be displayed. If the rule was created with the log checkbox enabled, the log will be generated even if the virtual service does not have full client logs enabled on the **Analytics** tab for the virtual service. These logs will still require **Non-Significant Logs** to be selected in order to be displayed (unless they qualify as **Significant Logs**).<br>■ **Significance**: If the connection or request is determined to be an error, it will be marked as `Significant`. This field provides a description of the issue (such as client terminated the connection or server returned `500 error`). |

| Field | Description |
| --- | --- |
| **Server and App data** | The third column provides the following information on connection or request and response:<br>■ **Server IP**: Pool name, server name, and the server IP address and port.<br>■ **Host**: The HTTP Host header, such as www.avinetworks.com or 10.1.1.10.<br>■ **Request**: The HTTP method (such as GET), version (such as HTTP/1.1), and size of the request (such as 2 Kb).<br>■ URl: The HTTP path and query of the client request.<br>■ **User Agent**: The raw client HTTP User-Agent header (such as Mozilla/5.0, AppleWebKit/533, and so on).<br>■ **Content Type**: The HTML, images, Javascript, and so on returned to the client.<br>■ **Response Length**: The size of the HTTP header plus content returned from the server to theNSX Advanced Load Balancer. (This might be different from the size of the response length from the NSX Advanced Load Balancer to the client due to compression, inserting JavaScript, or other acceleration that might alter the content size before it is sent to the client.) |
| **View All Headers** | Expands the log display to show additional information for the transaction. **View All Headers** exist due to the following:<br>■ **All Headers**: On the **Analytics** tab for the virtual service, create a new filter with the **All Headers** option selected. This will cause the NSX Advanced Load Balancer to record all client request and server response headers. Custom headers, cache control, and other useful troubleshooting can be done by viewing full headers. Larger headers come at the cost of a significant resource hit to the SEs creating the logs and the Controller storing the larger logs. Recommendation is to turn this feature on selectively, such as for specific clients or for a shorter time duration.<br>■ **DataScript Errors**: Many DataScript errors are caught when attempting to save a new script. However, there are many scenarios when the script could fail when executing. When this happens, an error will be created in the logs, visible under **View All Headers**. The error and stack trace are included to help determine the cause of the error. |

## Significant Log Events

For more information on the types of significant log events, see Significant Log Events.

# Application Troubleshooting 4

Application Troubleshooting explains the procedure to troubleshoot the NSX Advanced Load Balancer.

This chapter includes the following topics:

- Packet Capture
- Troubleshooting Packet Latencies within SE
- Tech Support

## Packet Capture

This section explains the use of packet capture to troubleshoot the NSX Advanced Load Balancer.

### Overview

Packet capture in the NSX Advanced Load Balancer runs a `TCPdump` for the designated virtual service or SE and provides complete visibility into the packet transmission.

Virtual services might be on a single SE or scaled out across multiple active SEs. The traffic captures are automatically executed on all SEs actively handling traffic for a virtual service. After the capture, the SE will forward the `PCAP` file to the Controller, which aggregates and sorts the client and server data into a single file.

**Note** It is highly recommended to set a limit for the capture which can either be the maximum number of packets to receive or the capture duration in minutes. On reaching the limit, the capture will be terminated and sent to the Controller.

The following are the traffic capture feature list:

- Capturing Virtual Service Traffic using CLI/UI
- Capturing SE Traffic using CLI
- SE PCAP Types
- Configurable Parameters for Virtual Service and PCAP
- SE PCAP File Rotation

- Downloading Packet Capture for SE PCAPS

## Capturing Virtual Service Traffic using CLI/UI

This section explains on how to capture the virtual service traffic using CLI/UI.

### Capturing Virtual Service Traffic using UI

- Navigate to **Operations > Traffic Capture**. The **Capture Configuration** section displays the parameters defined for current captures.
- Click the pencil icon to start a new capture.

In the **Traffic Capture** pop-up window, select the following based on your requirement:

| Field Name | Action |
|---|---|
| **Select Virtual Service** | Choose the virtual service for which you want to capture traffic. |
| **All Traffic** | Capture all traffic |
| **Choose Client IP, IP Range, Subnet Mask** | Capture traffic only for the specified IP address, list or range of IP addresses, or subnet.The IP addresses can be client or server addresses.<br>**Note**<br>■ To specify a list, use a space between each address. For example: 10.1.1.1 10.1.1.99 192.168.8.200.<br>■ To specify a range, use the following format: 10.1.1.1-10.1.1.255.<br>■ To specify a subnet, use the following format: 10.1.1.1/24 |
| **Number of Packets per Core** | Specify the maximum number of packets to capture in the core. |
| **Duration** | Specify the time in minutes to run the capture. |
| **Size of Packets** | Specify the size of the packet, in bytes, to be captured. This is similar to the snaplen option in TCPdump. To capture the entire packet, enter **0**. |
| ■ Under the **Advanced Settings** section, select one of the following options:<br>■ **Health monitor and data**<br>■ **Health monitor only**<br>■ **Data only** | Control the captures for health monitor flows. |
| **Capture Session Key** | Enable session key capture |

- Click **Start Capture** to view the progress of the capture.

**Note** For more information on enabling session key capture, see Enabling Session Key Capture When Debugging a Virtual Service.

## Completed Captures

On completing the capture, the Controller collates data from multiple SEs and formats the data into a `PCAP` file. These captures are then displayed under the **Completed Captures** section. It is tabluated as **Date**, **Virtual Service Name**, and **Size of Packets captured** which can be exported by downloading them in the `pcap` format. The capture file can be viewed using any standard traffic capture utility.

## Capturing Virtual Service Traffic using CLI

To capture packets using the NSX Advanced Load Balancer CLI, log into the shell prompt and enter the packet capture sub-mode for the desired virtual service:

```
debug virtualservice Test-virtual service
Updating an existing object. Currently, the object is:
+-------+-----------------------+
| Field |        Value          |
+-------+-----------------------+
| uuid  | virtualservice-0-1    |
| name  | Test-virtual service  |
+-------+-----------------------+
```

Parameters for packet capture are mentioned below. By default, the capture is performed within the context of the selected virtual service. It is also performed on all NSX Advanced Load Balancer SEs that are handling the virtual service traffic and includes the packets from the client and server side of the SE.

| Parameter | Definition |
|---|---|
| `capture_params duration` | Time, in minutes. Default is unlimited |
| `capture_params num_pkts` | Maximum number of packets to collect. Default is unlimited |
| `capture_params pkt_size` | Packet size, or snap length, to capture. Default is unlimited |
| `debug_ip addrs` | IPv4 address format |
| `debug_ip prefixes` | IPv4 prefix format <x.x.x.x/x> |
| `debug_virtual service_hm_include` | Include health monitor packets in the capture |
| `debug_virtual service_hm_none` | Omit health monitor packets from the capture (the default) |
| `debug_virtual service_hm_only Capture` | Only health monitor packets |

The `debug_ip` command enters a sub-mode. This allows multiple IP addresses or IP subnets to be entered. Omit the `debug_ip` option for subsequent entries. Save to commit the desired IPs and return to the previous menu.

**Note** By default, no maximum packets or duration of time to be captured are defined. It is recommended to include a maximum packet capture . Without a limit, the capture will run until the NSX Advanced Load Balancer SE drive is full, potentially disrupting service.

Specify parameters, including the maximum number of packets to capture:

```
debugvirtualservice> capture_params num_pkts 1000
debugvirtualservice> debug_ip addrs 10.10.10.10
debugvirtualservice:debug_ip> save
```

Begin capturing based on the previously configured parameters:

```
debugvirtualservice> capture
debugvirtualservice> save
+----------------+--------------------+
| Field          |          Value     |
+----------------+--------------------+
| uuid           | virtualservice-0-1 |
| name           | Test-VS            |
| debug_ip       |                    |
| addrs[1]       | 10.10.10.10        |
| capture        | True               |
| capture_params |                    |
| duration       | 0 mins             |
| num_pkts       | 1000               |
+----------------+--------------------+
```

Re-enter the packet capture sub-mode and stop an ongoing packet capture:

```
debug virtualservice Test-virtual service
debugvirtualservice> no capture
debugvirtualservice> save
```

## Capturing Virtual Service Packet in PCAPng Format

Virtual service packets can be captured in `PCAPng` format. Each packet contains the transmission direction , se-uuid, and core number that processed it. The frontend and backend flows are provided with a unique flow ID that allows you to co-relate the frontend and backend connections in the packet captures without going back to connection/app logs.

**Note** The flow ID might not be present for all packets in the flow.

To turn off this feature and fallback to `PCAP`, you can use the following command:

```
debug virtualservice <>
capture_params
no pcap_ng
```

### Exporting Packet Capture for Virtual Service PCAPs

Export the packet capture to a remote system that can view it using a tool like TCPdump or Wireshark:

```
show debug virtualservice Test-virtual service capture
Please specify the destination directory: /tmp
Downloaded the attachment to /tmp/virtual service_virtualservice.20141205_192033.pcap
bash
```

```
scp /tmp/virtual service_virtualservice.192033.pcap user@10.1.1.1:/tmp
```

## Capturing SE Traffic using CLI

This section describes the procedure to capture SE traffic using CLI.

Login to the shell using NSX Advanced Load Balancer CLI and then enter into the packet capture sub-mode for the SE.

Start the packet capture using the following `debug serviceengine <SE IP address>` command:

```
[admin:cntrl1]: > debug serviceengine 10.10.22.107
Updating an existing object. Currently, the object is:
+------------+--------------------------+
| Field      | Value                    |
+------------+--------------------------+
| uuid       | se-10.10.22.107-avitag-1 |
| name       | 10.10.22.107             |
| tenant_ref | admin                    |
+------------+--------------------------+
```

## SE PCAP Types

This section describes the types of `SE PCAP`s to enable packet capture accordingly.

To start packet capture for an NSX Advanced Load Balancer SE, use the `debug serviceengine <SE IP address>` command.

The various types of `SE PCAP`s are tabluated below:

| SE PCAPs | Description | Minimum Supported Version |
|---|---|---|
| SE level PCAP | Captures every ingress and egress packets received/ sent by a given SE.<br>File format: `se_core_<core_num>.pcap`<br>SE debug flag: `debug_pcap_rx`, `debug_pcap_tx` | 17.2.12 |
| Dropped PCAP | Packets dropped by SE are captured and stored into pcap file. Reason for drop is stamped on the comment section of the packet.<br>File format: `se_dropped_<core_num>.pcapng`<br>**Note** Few dropped packets might not be present in this capture file but this gives some level of visibility.<br>SE debug flag: `debug_pcap_drop` | 18.2.5 |
| IPC PCAP | IPC messages exchanged between SE's are captured. Type of IPC message is stamped on comment section of the packet.<br>File format: `se_ipc_<core_num>.pcapng`<br>SE debug flag: `debug_pcap_se_ipc` | 18.2.5 |
| DOS PCAP | DOS packets are captured.<br>File format: `se_dos_<core_num>.pcap`<br>SE debug flag: `debug_pcap_dos` | 17.2.12 |
| NAT PCAP | Captures all the NATed traffic on a given SE. Packet comments help in determining if a flow is external/ internal.<br>File format: `se_nat_<core_num>.pcapng`<br>SE debug flag: `debug_pcap_nat` | 21.1.1 |
| ROUTING PCAP | Capture all the packets that are routed by a given SE.<br>File format: `se_routing_<core_num>.pcapng`<br>SE debug flag: `debug_pcap_routing` | 21.1.1 |

**Note** Capture knob under `debug serviceengine <SE IP address>` command will help in enabling all types of captures at once. Since, this starts all the SE captures, disk memory might be consumed.

# Configurable Parameters for Virtual Service and PCAP

This section explains the methods to to configure parameters for virtual service and `PCAP`.

## Filtering based on IP Address

Execute the `debug_ip addrs <IP address for filter>` command from the `debugserviceengine` mode to filter the SE packet capture for the specific IP address.

```
[admin:cntrl1]: debugserviceengine> debug_ip addrs 10.10.10.10
[admin:cntrl1]: debugserviceengine:debug_ip>
[admin:cntrl1]: debugserviceengine:debug_ip> save
[admin:cntrl1]: debugserviceengine> where
Tenant: admin
+----------------+--------------------------+
| Field          | Value                    |
+----------------+--------------------------+
| uuid           | se-10.10.22.107-avitag-1 |
| name           | 10.10.22.107             |
| debug_ip       |                          |
|   addrs[1]     | 10.10.10.10              |
| tenant_ref     | admin                    |
+----------------+--------------------------+
```

## Filtering based on Capture Duration

Execute the `capture_params duration <duration in minutes>` command from the `debugserviceengine` mode to capture packets for the specified duration.

```
[admin:cntrl1]: debugserviceengine> capture_params duration 10
 [admin:cntrl1]: debugserviceengine> where
 Tenant: admin
 +----------------+--------------------------+
 | Field          | Value                    |
 +----------------+--------------------------+
 | uuid           | se-10.10.22.107-avitag-1 |
 | name           | 10.10.22.107             |
 | debug_ip       |                          |
 |   addrs[1]     | 10.10.10.10              |
 | capture_params |                          |
 |   duration     | 10 min                   |
 | tenant_ref     | admin                    |
 +----------------+--------------------------+
```

## Filtering based on Capture Packet Size

Execute the `capture_params pkt_size <packet size in bytes>` command from the `debugserviceengine` mode to capture traffic of the desired packet size.

```
[admin:cntrl1]: debugserviceengine> capture_params pkt_size 0
[admin:cntrl1]: debugserviceengine> where
Tenant: admin
```

```
+----------------+--------------------------+
| Field          | Value                    |
+----------------+--------------------------+
| uuid           | se-10.10.22.107-avitag-1 |
| name           | 10.10.22.107             |
| debug_ip       |                          |
|   addrs[1]     | 10.10.10.10              |
| capture_params |                          |
|   pkt_size     | 0 bytes                  |
|   duration     | 10 min                   |
| tenant_ref     | admin                    |
+----------------+--------------------------+
[admin:cntrl1]: debugserviceengine> save
[admin:cntrl1]:
```

The above packet filter captures traffic for the Service Engine `10.10.22.107` with the IP address `10.10.10.10` for a duration of 10 minutes with packet size `0`.

To stop the ongoing packet capture, re-enter the packet capture sub-mode and execute `no capture` command.

```
[admin:cntrl1]: debug service engine 10.10.22.107
[admin:cntrl1]: debugserviceengine> no capture
[admin:cntrl1]: debugserviceengine> save
```

Export the packet capture to a remote system and use tools like TCPdump or Wireshark for further analysis.

For more information on exporting packet capture, see Exporting Packet Capture for Virtual Service PCAPs.

## Configurable Size for both Virtual Service and SE PCAPs

The following are the configurable size for both virtual service and SE `PCAP`s:

- The maximum `PCAP` file size per SE can be specified in this section. Each core gets configured (maximum size)/(total number of cores).
- `absolute_size`: The maximum `PCAP` file size in MegaBytes.
- `percentage_size`: The percentage of free disk space available in NSX Advanced Load Balancer partition. The valid range is 0-75%. The free size is instantaneous free size at the time of configuration. This should be used with caution.
- Set `absolute_size` and `percentage_size` to `0` to use NSX Advanced Load Balancer's default maximum allowed `PCAP` size (100MB) per core.
- The maximum of `absolute_size` and `percentage_size` will be considered as maximum `PCAP` file.

```
[admin:ctrl]: > debug serviceengine 10.10.100.10
[admin:ctrl]: debugserviceengine> capture_params
[admin:ctrl]: debugserviceengine:capture_params> capture_file_size
```

```
[admin:ctrl]: debugserviceengine:capture_params:capture_file_size> absolute_size 500
[admin:ctrl]: debugserviceengine:capture_params:capture_file_size> save
[admin:ctrl]: debugserviceengine:capture_params> save
[admin:ctrl]: debugserviceengine> save
```

**Note**

- You need to observe caution while using `percentage_size` value. It is the instantaneous free size which is taken into consideration.
- With virtual service and `SE PCAP` configured with `percentage_size`, the cumulative number of these sizes during configuration is not supported. This could lead to disk space problems. Excessive enabling of debugs could also aggravate the same.

## SE PCAP File Rotation

This section explains the properties and commands of SE PCAP file rotation.

The following are the properties of SE level `PCAP` file rotation:

- `file_count` under capture param knob can be used to turn on file rotation of the pcap file.
- `file_count` greater than one indicates that the file rotate is on.
- With the file rotate on, the latest `configured_pcap_file_size` of data will be present in the captured file.
- With the file rotate off, `configured_pcap_file_size` of data from the time of capture start will be captured.

The following are the CLI commands under SE PCAP file rotation:

```
[admin:ctrl]: > debug serviceengine 10.19.108.10
[admin:ctrl]: debugserviceengine> capture_params
[admin:ctrl]: debugserviceengine:capture_params> file_count 2
[admin:ctrl]: debugserviceengine:capture_params> save
[admin:ctrl]: debugserviceengine> save
```

### Downloading Packet Capture for SE PCAPs

You can gather `SE PCAP`s from the Controller from `/var/lib/avi/se-pcap` path. Use the following code to download packet capture for `SE PCAP`s:

```
scp /var/lib/avi/se-pcap/se*.pcap user@10.1.1.1:/tmp
```

# Troubleshooting Packet Latencies within SE

This section explains the steps to configure analytics profile to trouble

SE time flow tracker can track the network characteristics, processing time at critical checkpoints, and flag queuing delays in a packet journey through the network appliance.

# Configuring Analytics Profile through CLI

The following are the configuration used in analytics profile:

```
> show analyticsprofile System-Analytics-Profile
..
| latency_audit_props                |                           |
|   latency_audit_mode               | LATENCY_AUDIT_OFF         |
|   latency_threshold                | 20 milliseconds           |
|   conn_est_audit_mode              | LATENCY_AUDIT_ON          |
|   conn_est_threshold               | 40 milliseconds           |
+------------------------------------+---------------------------+
```

The audit properties are described below:

| Audit Properties | Default | Description |
|---|---|---|
| latency_audit_mode | LATENCY_AUDIT_OFF | LATENCY_AUDIT_OFF - Default, no latency audit is performed.<br>LATENCY_AUDIT_ON - Turn on the latency audit with statistics/ counters for flows/ packets breaching the configured threshold.<br>LATENCY_AUDIT_ON_WITH_SIG - Turn on the latency audit, statistics are updated along with event and significant logs. |
| latency_threshold | 20 msec | This enables tracking the dispatcher to proxy latency for each packet if latency_audit_mode is set to LATENCY_AUDIT_ON. This is the threshold above which events, significant logs and metrics are expressed if the per packet latency from dispatcher to proxy is too high. |

| Audit Properties | Default | Description |
|---|---|---|
| conn_est_audit_mode | LATENCY_AUDIT_ON | LATENCY_AUDIT_OFF -No connection establishment audit is performed.<br>LATENCY_AUDIT_ON - Default, turn on the connection establishment audit with statistics/ counters for flows/ packets breaching the configured threshold.<br>LATENCY_AUDIT_ON_WITH_SIG - Turn on the connection establishment audit, statistics are updated along with event and significant logs. |
| conn_est_threshold | 40 msec | This enables tracking the TCP connection establishment time if conn_est_audit_mode is set to LATENCY_AUDIT_ON. This is the threshold for anomaly detection which is expressed as events, significant logs and metrics if this threshold is breached.se |

**Note** Currently, latency_audit_filters is supported only for TCP/IPV4.

## Configuring latency_audit_filters in debug Virtual Service

The filters contain all the options offered by VS capture filters. However, latency_audit_filters are functionally independent of capture filters.

```
> debug virtualservice vs-1
..
[admin:vpr-ctrl1]: debugvirtualservice:latency_audit_filters>
cancel               Exit the current submode without saving
capture_ip           (submode)
capture_ipc          (submode)
do                   Execute a show command
dst_port_end         Destination Port range filter.
dst_port_start       Destination Port range filter.
eth_proto            Ethernet Proto filter.
ip_proto             IP Proto filter. Support for TCP only for now.
new                  (Editor Mode) Create new object in editor mode
no                   Remove field
save                 Save and exit the current submode
show_schema          show object schema
src_port             Source Port filter.
src_port_range_end   Source Port range end filter. If specified, the source port filter will
be a range. The filter range will be between src_port and src_port_range_end.
tcp_ack              TCP ACK flag filter.
tcp_fin              TCP FIN flag filter.
tcp_push             TCP PUSH flag filter.
tcp_syn              TCP SYN flag filter.
```

```
watch                Watch a given show command
where                Display the in-progress object
```

## Metrics and Logs

The framework supports metrics, events and logs which can be configured as follows:

Metrics at SE level

```
raised request
```

Metrics at Virtual Service level

```
raised request
```

Events

<<Raised image Request>>

Note: The threshold is set to 0 in this example.Significant Logs (When Latency_Audit is enabled).

Significant Logs (When Latency_Audit is enabled)

<<Raised image Request>>

The detailed timing and flow characteristics will be present in Connection/App Log.

# Tech Support

This section explains the proactive tech support services offered by NSX Advanced Load Balancer Pulse.

## Proactive cases

This section describes the steps to create and debug proactive cases.

### Creating Proactive Tech-Support Cases

With proactive cases, the NSX Advanced Load Balancer Controller will create a customer case whenever a critical event occurs in the system such as a SE process crash or a Controller process crash. It automatically collects the relevant files like core archive, tech-support bundles and attaches them to the created support case with zero user intervention.

There are two opt-ins available for proactive cases:

1 enable_auto_case_creation_on_controller_failure

2 enable_auto_case_creation_on_se_failure

You can use the following CLI configuration to enable these opt-ins:

```
admin:ctrl1]: albservicesconfig> case_config
[admin:ctrl1]: albservicesconfig:case_config> enable_auto_case_creation_on_controller_failure
Overwriting the previously entered value for enable_auto_case_creation_on_controller_failure
```

```
admin:ctrl1]: albservicesconfig:case_config> enable_auto_case_creation_on_se_failure
Overwriting the previously entered value for enable_auto_case_creation_on_se_failure
admin:ctrl1]: albservicesconfig:case_config> save
[admin:ctrl1]: albservicesconfig> save
```

Once either of the opt-in options are selected, the system will enable the alert configuration which monitors the Audit Compliance Event.

You can view the proactive configuration by navigating to **Operations > Alerts> > Alert Config**.

## Proactive Case Creation Workflow

The NSX Advanced Load Balancer Controller monitors for failure events/ alerts. After the failure events occur, the proactive case creation service creates a case and attaches the tech support/ core to the created case based on the failures.

## Debugging Proactive Tech support Issues

You can debug proactive tech support issues in two simple steps:

1 Ensure that a core event is raised.

2 Check the contents of the **Action Script Output** section of the generated alert once it is raised.