# Configuring SAN Analytics

This chapter provides information about the SAN Analytics feature and how to configure it:

# Feature History for Configuring SAN Analytics

*Table 1: Feature History for Configuring SAN Analytics*

| Feature Name | Release | Feature Information |
|---|---|---|
| SAN Analytics | 8.3(1) | The following commands were introduced:<br>• **analytics port-sampling module** *number* **size** *number* **interval** *seconds*<br>• **show analytics port-sampling module** *number*<br>• **no analytics name** *query_name*<br><br>Refer to the Table 3: Command Changes, on page 8 for commands that were changed from Cisco MDS NX-OS Release 8.2(1) to Cisco MDS NX-OS Release 8.3(1). |
| Port Sampling | 8.3(1) | The Port Sampling feature allows you to gather data from a subset of ports in a module that is being monitored, cycle through the subset of ports, and stream data from these ports at a regular port sampling interval. |
| SAN Analytics | 8.3(1) | The following flow metrics were introduced:<br>• read_io_scsi_busy_count<br>• read_io_scsi_check_condition_count<br>• read_io_scsi_queue_full_count<br>• read_io_scsi_reservation_conflict_count<br>• sampling_end_time<br>• sampling_start_time<br>• total_time_metric_based_read_io_bytes<br>• total_time_metric_based_read_io_count<br>• total_time_metric_based_write_io_bytes<br>• total_time_metric_based_write_io_count<br>• write_io_scsi_busy_count<br>• write_io_scsi_check_condition_count<br>• write_io_scsi_queue_full_count<br>• write_io_scsi_reservation_conflict_count<br><br>For more information, see Flow Metrics. |

| Feature Name | Release | Feature Information |
|---|---|---|
| SAN Analytics Support for Cisco MDS 9132T 32-Gbps 32-Port Fibre Channel Switch | 8.3(1) | Added the Cisco MDS 9132T 32-Gbps 32-Port Fibre Channel switch to the list of supported hardware. |
| SAN Analytics | 8.2(1) | Added the Cisco MDS 9700 48-Port 32-Gbps Fibre Channel Switching module to the list of supported hardware. |
| SAN Analytics Support for Cisco NPV switches | 8.3(1) | Added guidelines and limitations for using the SAN Analytics feature on Cisco NPV switches. |
| SAN Analytics | 8.2(1) | The SAN Analytics feature allows you to monitor, analyze, identify, and troubleshoot performance issues on Cisco MDS 9000 Series Multilayer Switches.<br>The following commands have been introduced:<br>• **analytics type fc-scsi**<br>• **analytics query** *"query_string"* **type timer** *timer_val*<br>• **clear analytics** *"query_string"*<br>• **feature analytics**<br>• **purge analytics** *"query_string"*<br>• **ShowAnalytics**<br>• **show analytics** {**query** {*"query_string"* \| *id* **result**} \| **type fc-scsi flow congestion-drops** [**vsan** *number*] [**module** *number* **port** *number*]} |

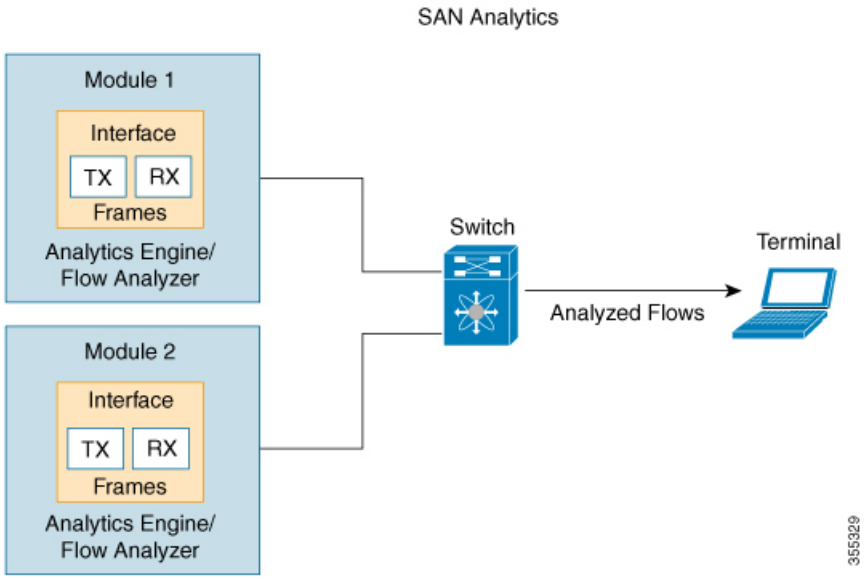
# SAN Analytics Overview

The SAN Analytics feature allows you to monitor, analyze, identify, and troubleshoot performance issues on Cisco MDS switches. For a list of supported switches, see the Hardware Requirements for SAN Analytics, on page 5.

In a Fibre Channel SAN environment, it is important to provision, monitor, and maintain the performance of all devices to be able to resolve any issues that can hinder the performance of such devices. The SAN Analytics feature monitors flows bidirectionally, correlates the flows in a network processing unit (NPU) within a module, and provides the fully analyzed network data to the user.

The following figure shows the functionality of the SAN Analytics feature:

***Figure 1: SAN Analytics Overview***




The SAN Analytics feature helps you to overcome the following issues in the SAN Analytics solutions:

- This feature defines a flow as bidirectional series of packets; every input and output flow is analyzed in the module before the fully analyzed data is exported for analysis.
- This feature does not require any redesign, disruption, or additional hardware for an existing network.
- This feature eliminates the need for optical Traffic Access Points (TAPs) and delivers near real-time data using which performance issues can be monitored, analyzed, and resolved instantly.

# Hardware Requirements for SAN Analytics

The following table lists the hardware requirements for the SAN Analytics feature:

***Table 2: List of Supported Hardware***

| Switch | Module |
|---|---|
| Cisco MDS 9700 Series Multilayer Directors | • Cisco MDS 9700 48-Port 32-Gbps Fibre Channel Switching Module (DS-X9648-1536K9) |
| Cisco MDS 9132T 32-Gbps 32-Port Fibre Channel Switch | • 16 x 32-Gbps Fixed Ports (DS-C9132T-K9-SUP)<br>• 16-Port 32-Gbps Fibre Channel Expansion Module (M9XT-FC1632) |

# Guidelines and Limitation for SAN Analytics

- This feature is not supported on VSANs where:
  - The default zone permit is configured.
  - The Inter-VSAN Routing (IVR) or Cisco MDS 9000 Input/Output Accelerator (IOA) feature is enabled.
  - The interoperability mode is enabled.
- This feature has the following restriction with regard to queries:
  - The maximum number of push queries is eight. For information about push queries, see Information About SAN Analytics, on page 9.
  - Does not support clearing and purging of individual metrics. For information about clearing and purging metrics, see Information About SAN Analytics, on page 9.
  - The where condition in the query syntax can accept only the equal (=) operator. For more information, see Query Syntax, on page 21.
- This feature does not analyze flow metrics for any non-Read or Write traffic.
- For this feature to work as documented, Network Time Protocol (NTP) must be synchronized between the switch, modules, and DCNM. For information on NTP, see the "Configuring NTP" section in the Cisco MDS 9000 Series Fundamentals Configuration Guide.
- This feature is not supported on Switched Port Analyzer (SPAN) Destination (SD) ports and NP ports. If you are enabling this feature on a range of interfaces, ensure that there are no SD or NP ports in that range. Otherwise, this feature does not get enabled on any interface.
- This feature only monitors Fibre Channel SCSI traffic at present. Support for the Fibre Channel Non-Volatile Memory Express (FC-NVMe) traffic will be added in a future release.
- If the **feature analytics** command is enabled in Cisco MDS NX-OS Release 8.2(1) or Cisco MDS NX-OS Release 8.3(1), then upgrading or downgrading between Cisco MDS NX-OS Release 8.2(1) and Cisco MDS NX-OS Release 8.3(1) is supported only after disabling this feature using the **no feature analytics** command before upgrading or downgrading and then enabling this feature using the **feature analytics** command.
- After upgrading, downgrading, reloading a switch, or reloading a module, all the flow metrics will be purged.
- We recommend that the steaming interval (**snsr-grp** *id* **sample-interval** *interval*), port sampling interval (**analytics port-sampling module** *number* **size** *number* **interval** *seconds*), and push query interval (**analytics query** *"query_string"* **name** *query_name* **type periodic** [**interval** *seconds*] [**clear**] [**differential**]) to be the same. Also, we recommend that you change the push query interval, port sampling interval, and streaming interval in the ascending order.
- We recommend that you set the streaming interval, port sampling interval, and push query interval to be equal to or more than the minimum recommended value of 30 seconds. Configuring intervals below the minimum value may result in undesirable system behavior.

- The maximum number of Initiator-Target-LUNs (ITLs) supported per module and switch with fixed module is 20,000.
- The metrics can be inaccurate when the network processing unit (NPU) load exceeds its capacity. In such a situation, use the Port Sampling feature to analyze the metrics. For more information, see Port Sampling, on page 12.
- After you purge a view instance and its associated metrics, we recommend that you wait for few seconds before executing a pull query. This is because some fields in the flow metrics may contain irrelevant values until the purge operation is complete.
- This feature tracks every flow metric on a per-port basis. Flow requests and responses spanning different physical ports on a switch may result in some flow metrics not being accurately computed. This specifically occurs when the SAN Analytics feature is enabled on Inter-Switch Link (ISL) ports (E ports).

  The following is a lists the scenarios where a request response can be seen on different ISL ports:

  - The load-balancing scheme is changed to Source ID (SID)-Destination ID (DID) by the user using the **vsan** *ID* **loadbalancing src-dst-id** command.
  - ISLs (E ports) are configured to nontrunking mode by the user using the **switchport trunk mode off** command.
  - ISLs (E ports) that are a part of a port channel, and the port-channel is not configured to the active mode using the **no channel mode active** command.
  - ISLs are not bundled together to be a part of a port channel.
  - There is a port channel between the Cisco MDS 9250i Multiservice Fabric Switch or Cisco MDS 9148S 16-G Multilayer Fabric Switch, and the Cisco MDS 9700 48-Port 32-Gbps Fibre Channel Switching Module (DS-X9648-1536K9).

# Command Changes

There were changes to some of the commands in Cisco MDS NX-OS Release 8.3(1) when compared to Cisco MDS NX-OS Release 8.2(1). This document uses commands introduced or changed in Cisco MDS NX-OS Release 8.3(1). Refer to the Table 3: Command Changes, on page 8 for the equivalent commands used in Cisco MDS NX-OS Release 8.2(1).

The recommended release for using the SAN Analtyics feature is Cisco MDS NX-OS Release 8.3(1).

Table 3: Command Changes, on page 8 provides the changes in the commands from Cisco MDS NX-OS Release 8.2(1) to Cisco MDS NX-OS Release 8.3(1):

***Table 3: Command Changes***

| Cisco MDS NX-OS Release 8.3(1) | Cisco MDS NX-OS Release 8.2(1) |
|---|---|
| **analytics query** *"query_string"* **name** *query_name* **type periodic** [**interval** *seconds*] [**clear**] [**differential**] | **analytics query** *"query_string"* **type timer** *timer_val* |
| **clear analytics query** *"query_string"* | **clear analytics** *"query_string"* |
| **purge analytics query** *"query_string"* | **purge analytics** *"query_string"* |
| **show analytics query** {*"query_string"* [**clear**] [**differential**] \| **all** \| **name** *query_name* **result**} | **show analytics query** {*"query_string"* \| *id* **result**} |

# Information About SAN Analytics

The SAN Analytics feature collects flow metrics using frames of interest, for data analysis, and includes the following components:

- Data Collection—The flow data is collected and stored in the SAN Analytics database on the switch supervisor.
- On-board Querying—The data that is stored in a database can be extracted using a pull query, a push query, or overlay CLIs. Queries are used to extract the frames of interest from the database. The frames of interest are used to monitor, analyze, and troubleshoot performance issues on a switch. For more information, see Constructing and Using Queries, on page 34.

  The following are the different ways in which to query the database:

  - Pull query—A one-time query used to extract the flow information that is stored in a database at the instant the query is executed. The output is in JSON format.

    Overlay CLI—A predefined pull query that displays the flow metrics in a user-friendly tabular format.

  - Push query—A recurring query installed to periodically extract the flow metrics that are stored in a database and send them to a destination. The output is in JSON format.

    From Cisco MDS NX-OS Release 8.3(1), the following options are available for a push query:

    - Clear—Clears all the minimum, maximum, and peak flow metrics after the streaming interval.
    - Differential—Streams only the ITL flow metrics that have changed between streaming intervals.

    Push query supports the following modes for extracting the flow metrics:

    - Continuous mode—Data is gathered continuously on all analytics enabled ports at a configured port sampling interval. For example, data is gathered continuously from all analytics enabled ports with a port sampling interval of 30 seconds.
    - Sampling mode—Data is gathered on a subset of the analytics enabled ports at a configured port sampling interval. For example, data is gathered on a group of 6 ports of the 24 analytics enabled port with a port sampling interval of 30 seconds. For more information. see Port Sampling, on page 12.

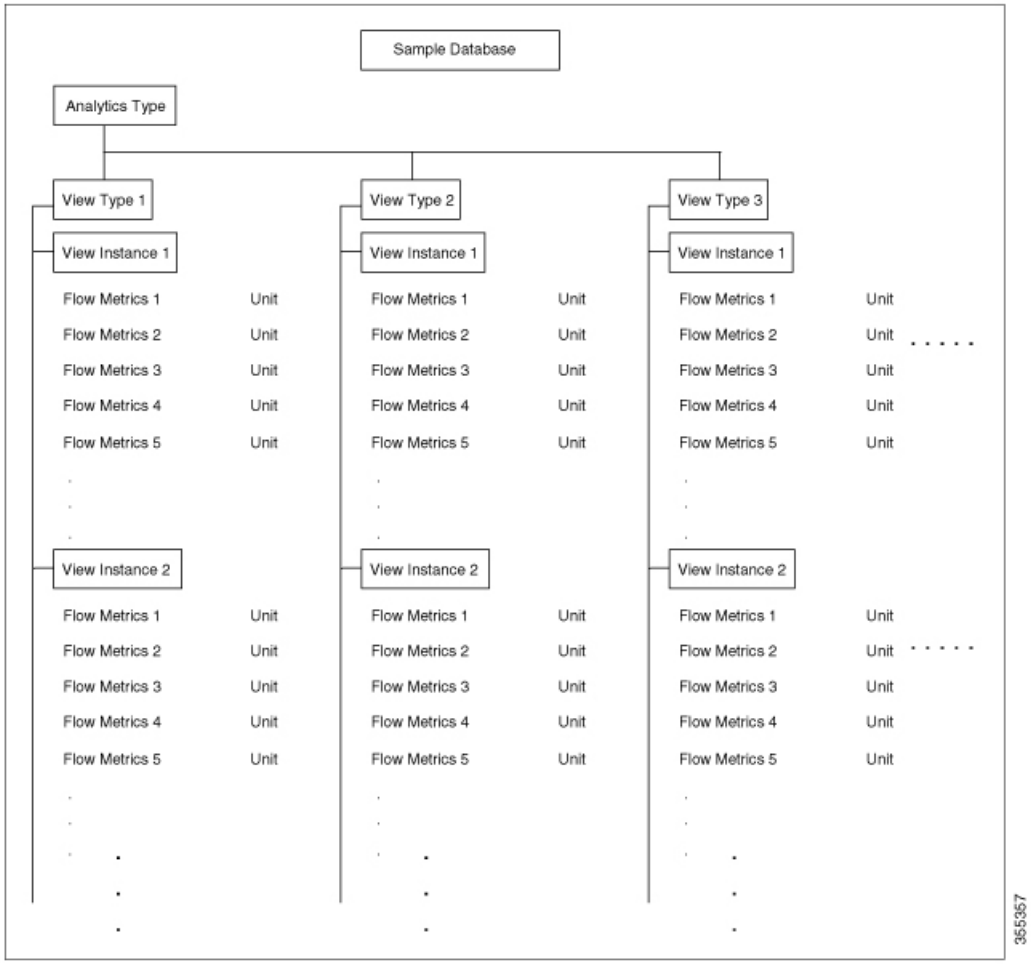
The database used for collecting and storing the flow metrics is organized according to the following hierarchy:

- Analytics Type—Specifies the analytics type. Note that only *fc-scsi* type is supported in Cisco MDS NX-OS Release 8.2(1) or later releases.
- View Type—Specifies the view type of the metric database. Views are defined based on components that constitute a flow, for example, port view, initiator_IT view, target_ITL view, and so on. The query syntax is used to run queries on a view type. The syntax supports only one query on a single view type. For a list of view types supported, see List of Supported View Types, on page 22.
- View Instance—Specifies an instance of a given view type. View instance has its own flow metrics. For example, for port view type, fc1/1 is one instance, fc1/2 is another instance, and so on.

- Flow Metrics—Specifies the flow metrics that are used for analysis. For information about the list of flow metrics supported, see Port View Instance.

The following image shows the various components of a sample database:

***Figure 2: Sample Database***



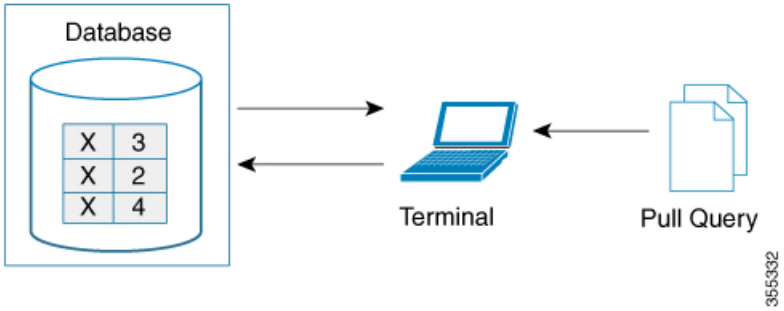

The following is the flow data collection workflow:

1. Feature Enablement—Enable the SAN Analytics feature on switches for which flow metrics have to be analyzed.
2. Interface Enablement—Enable collection of flow metrics on interfaces. We recommend that you enable the SAN Analytics feature on interfaces, as seen in the image in Deployment Modes, on page 14.
3. Executing and Installing Queries—The following queries are used to retrieve flow metrics from the database:
   - Pull Query—Provides near real-time flow metrics for troubleshooting issues directly on a switch. Data from a pull query is extracted from the database at that instance and responded to the query.

     Overlay CLI—A predefined pull query that displays the flow metrics in a user-friendly tabular format. It provides near real-time flow metrics for troubleshooting issues directly on a switch.

The following image shows the functionality of a pull query:

***Figure 3: Pull Query***



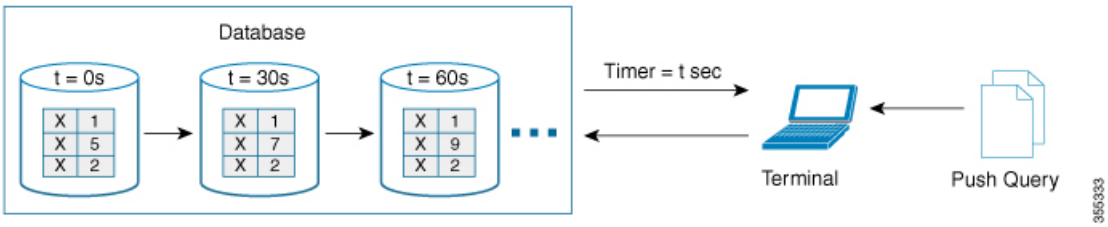

- Push Query—Provides flow metrics at regular intervals. You can specify a time interval, in seconds. After the time interval expires, the flow metrics that are of interest to the user are refreshed and pushed from the database. When multiple queries are installed, each of the push queries pushes the flow metrics independent of each other, which is the expected behavior.

**Note**

- Pull query, push query, and overlay CLI are applicable only on the interfaces on which the SAN Analytics feature is enabled.
- Push query timer fetches flow metrics from the supervisor and stores them in the database at a specified push query interval.

The following image shows the functionality of a push query where only certain metrics are set to be updated at specific intervals:

***Figure 4: Push Query***

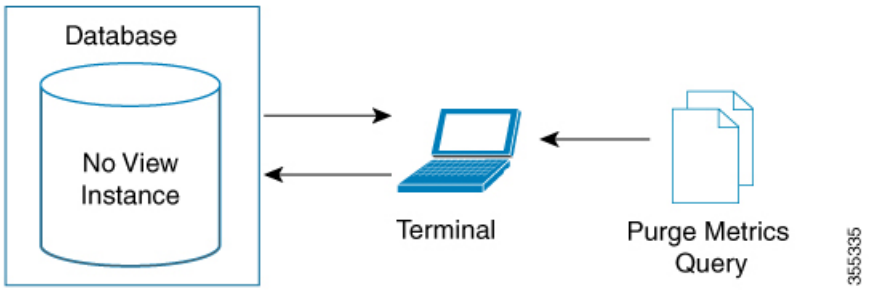



4. Clearing and Resetting Metrics—The following features allow you to clear or reset the flow metrics collected in a database:
   - Purge—Deletes a specified view instance and all the metrics associated with this view instance.

**Note** Purge deletes specific view instance and its associated metrics, whereas clear resets all the metrics of a view instance momentarily. After purging the database, the database will continue to collect the metrics for the specified *"query_string"*.

The following image shows the purge metrics query functionality:

***Figure 5: Purge Metrics Query***

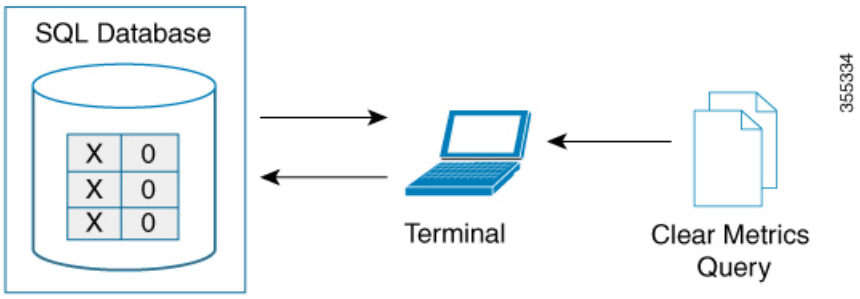



- Clear—Resets all the metrics that satisfy the query syntax specified in the **clear analytics query** *"query_string"* command.

**Note** Clear resets all metrics of a view instance, whereas purge deletes specific view instance and its associated metrics momentarily. After clearing the database, the database will continue to collect the metrics for the specified *"query_string"*.

The following image shows the clear metrics query functionality:

***Figure 6: Clear Metrics Query***




# Port Sampling

The Port Sampling feature introduced in Cisco MDS NX-OS Release 8.3(1) allows you to gather data from a subset of ports in a module that is already being monitored, cycle through the subset of ports, and stream data from these ports at a regular port sampling interval.

This feature is particularly useful when the NPU load is high and you cannot reduce the number of ports being monitored on a module. In such a situation, the load on the NPU can be reduced by sampling a subset of the monitored ports at a specified port sampling interval. Use the **show analytics port-sampling module** *number* command to check the NPU load.

Any IO and errors that occur on a monitored port while it is not being sampled are not seen and included in the analytics data.

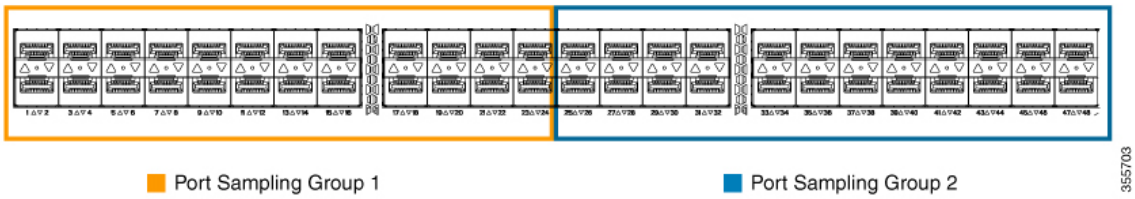
The port sampling interval used in this feature is independent of the streaming interval used in a pull query. That is, the port sampling interval need not start at the same time as the streaming interval. We recommend that you set the streaming interval, port sampling interval, and push query interval to be equal to or more than the minimum recommended value of 30 seconds.

**Note** When the Port Sampling feature is enabled on a module and later the SAN Analytics feature is enabled on new ports on the module, the port sampling data for the new ports will be streamed only after the next port sampling interval.

### Port Sampling Scenarios

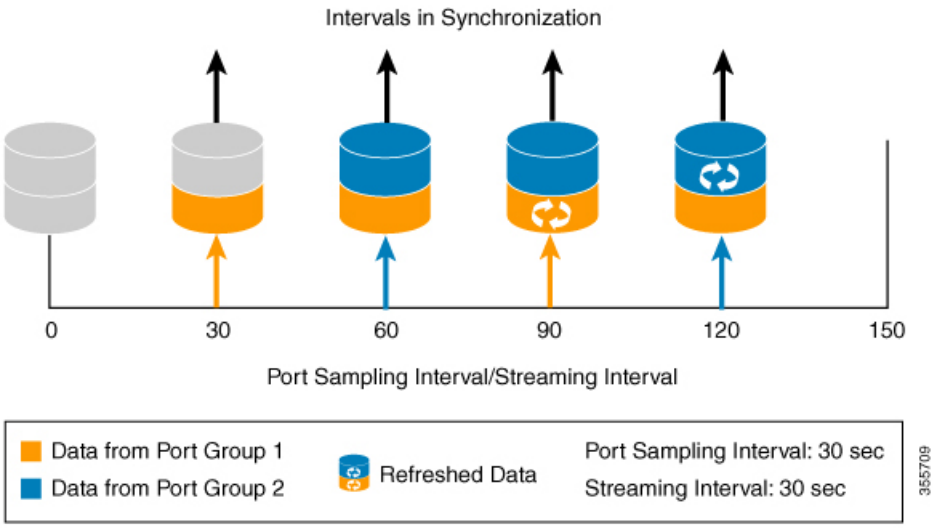
Let us consider a module consisting of 48 ports and group them into two subset of ports. Depending on the port sampling intervals configured for these subset of ports and the streaming interval configured, flow metrics can be captured at different intervals as seen in the examples below:

***Figure 7: Port Sampling Groups***



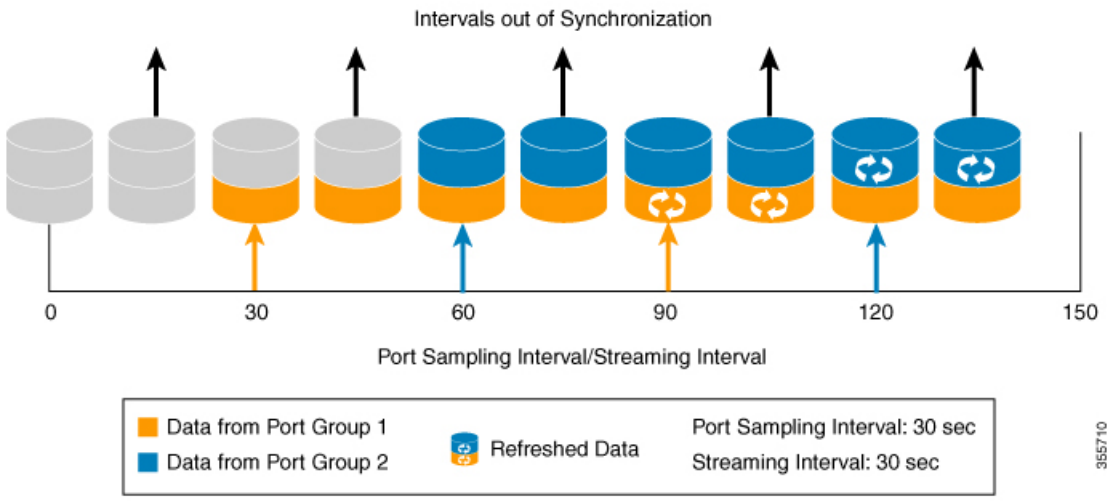

- When port sampling interval and streaming interval are in synchronization:

***Figure 8: Sampling Interval in Synchronization With the Streaming Interval***




- When port sampling interval and streaming interval are out of synchronization:

*Figure 9: Sampling Interval in out of Synchronization With the Streaming Interval*




# Deployment Modes

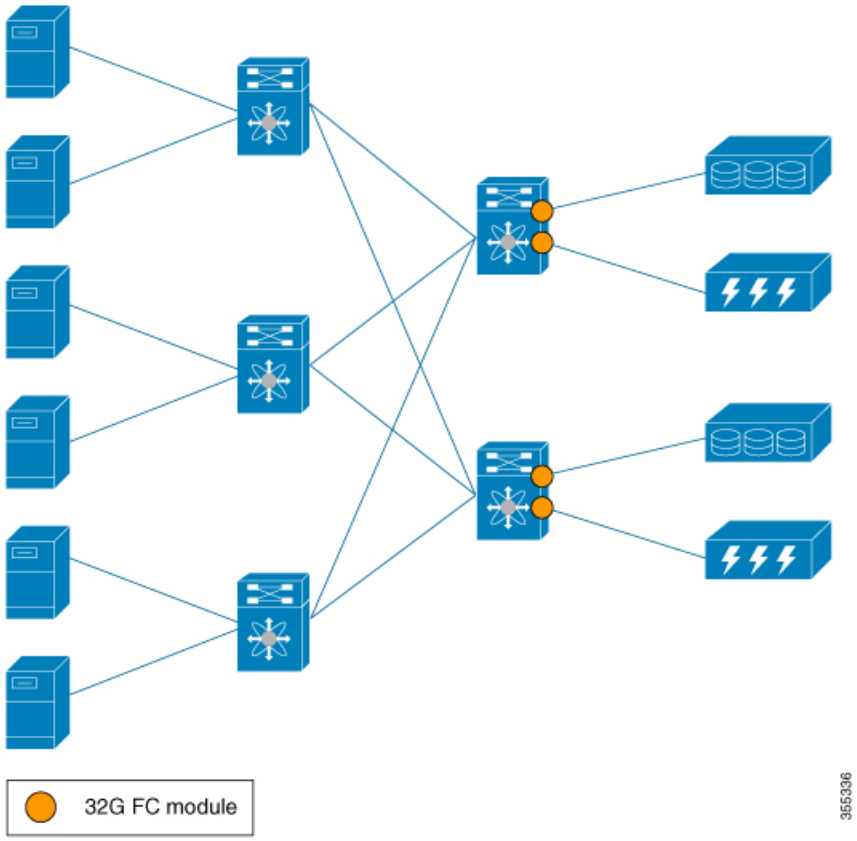
**Note** We recommend that you use the Host Edge or Storage Edge Deployment Mode because in the ISL and Global modes the traffic seen on an ISL port will vary based on the configuration changes in the fabric. This will result in a highly varying NPU load and makes it hard to determine if port sampling needs to be enabled and if enabled to determine the port sampling size.

Depending on where the switches that support the SAN Analytics feature are deployed in a SAN fabric, the following deployment modes are possible:

**Storage Edge Mode**

The SAN Analytics feature is enabled on all the Cisco MDS edge switches and on the interfaces that are connected to storage arrays.

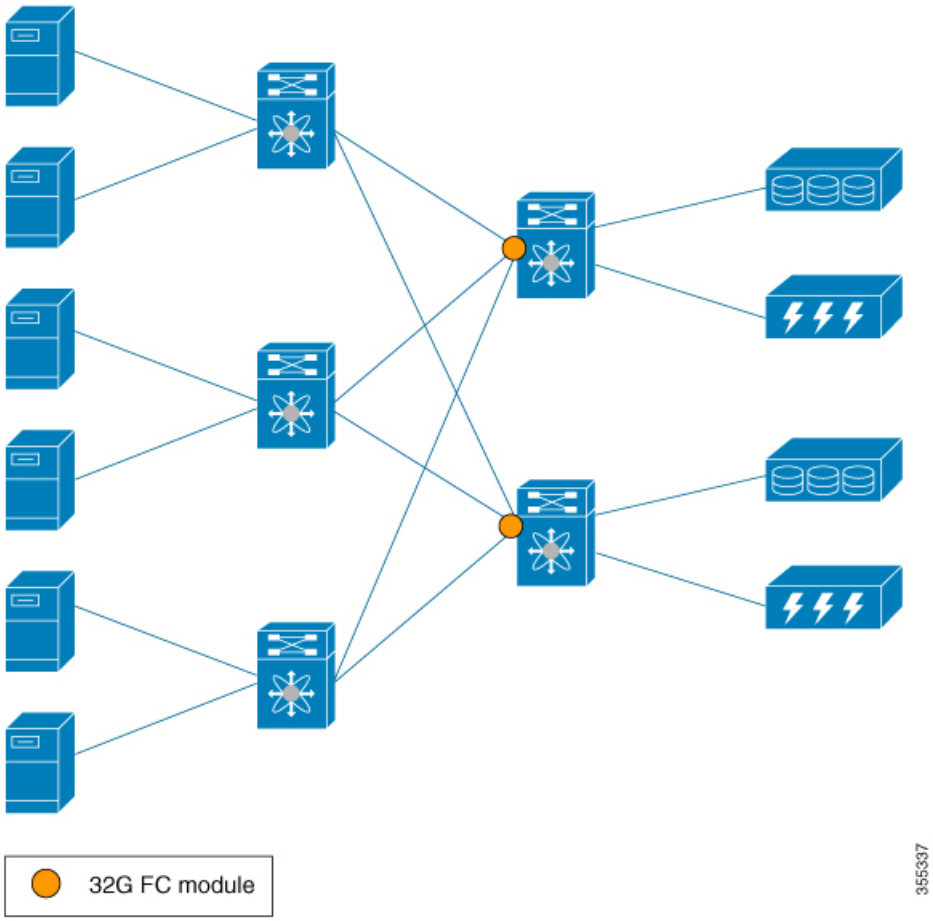
*Figure 10: Storage Edge Deployment Mode*




## ISL Mode

The SAN Analytics feature is enabled on all the Cisco MDS switches and on the interfaces that are on either side of ISLs.

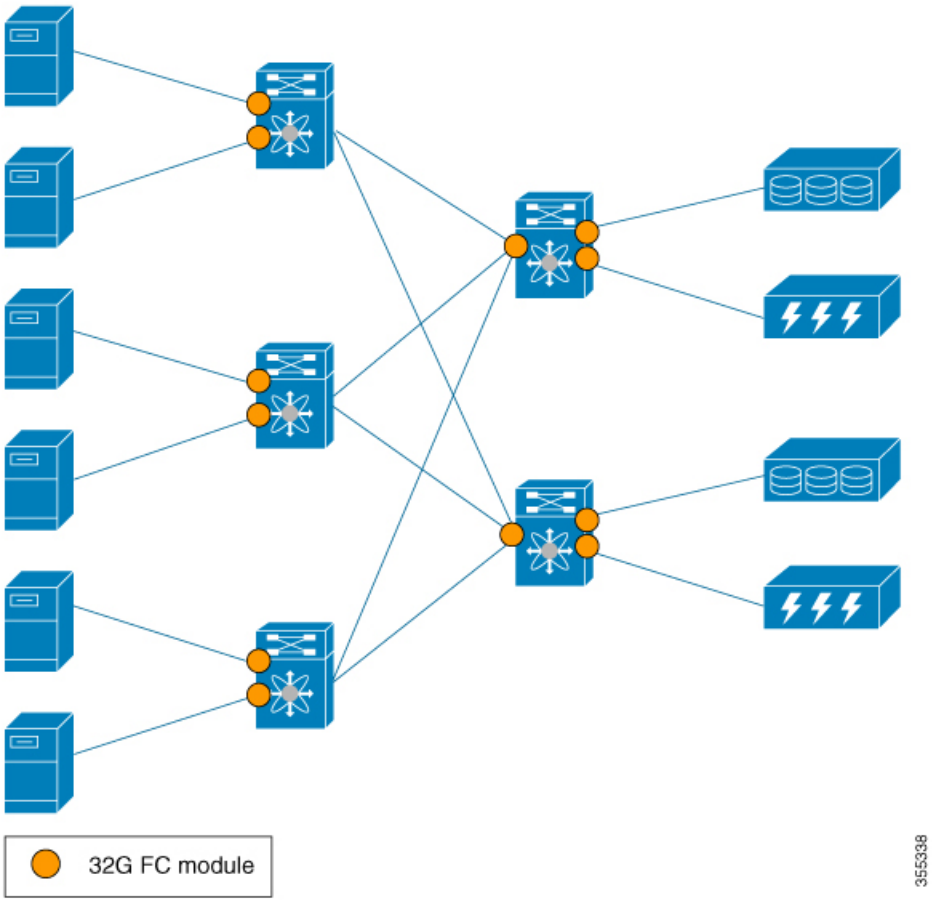
*Figure 11: ISL Deployment Mode*




## Global Mode

We recommended that you use the Global mode when there is a need for end-to-end visibility and correlation in a SAN fabric. The SAN Analytics feature is enabled on all the Cisco MDS switches and all the interfaces.

*Figure 12: Global Deployment Mode*

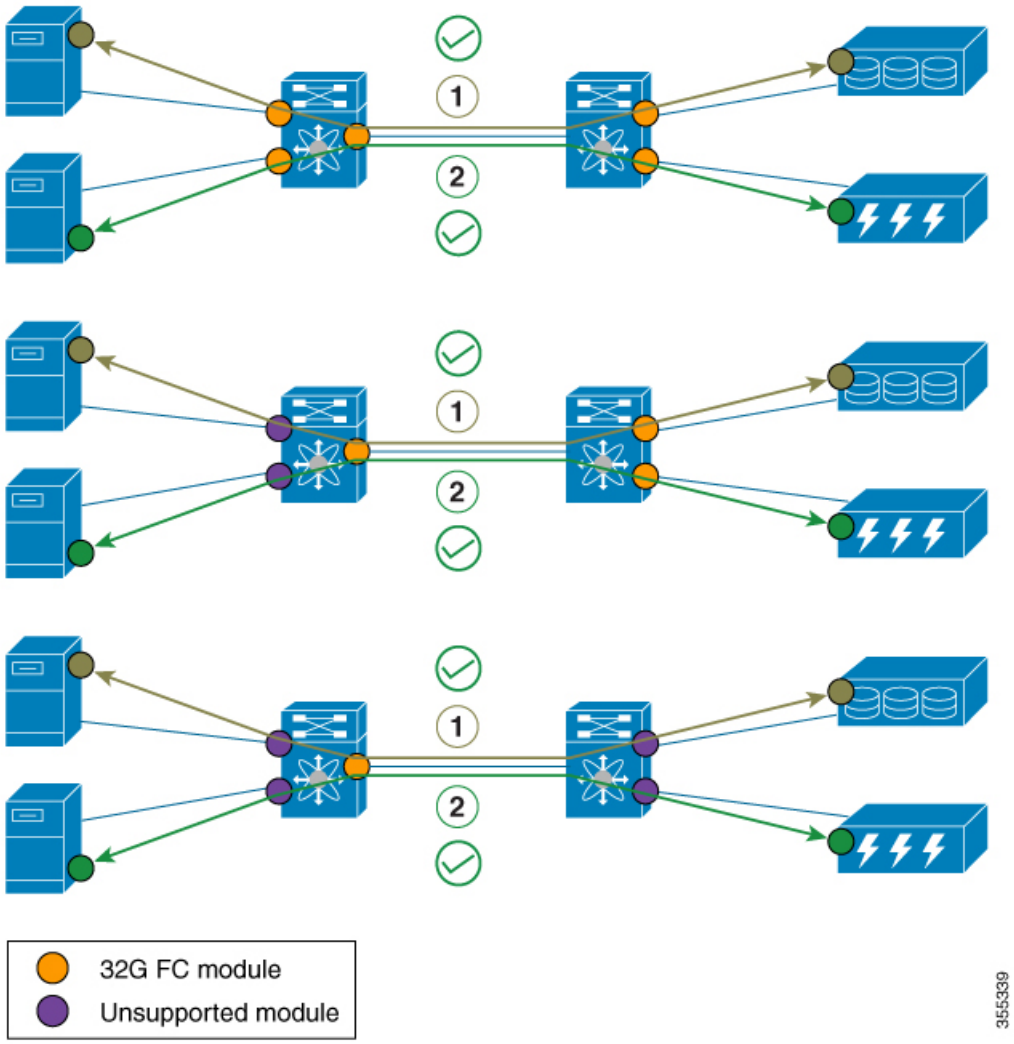


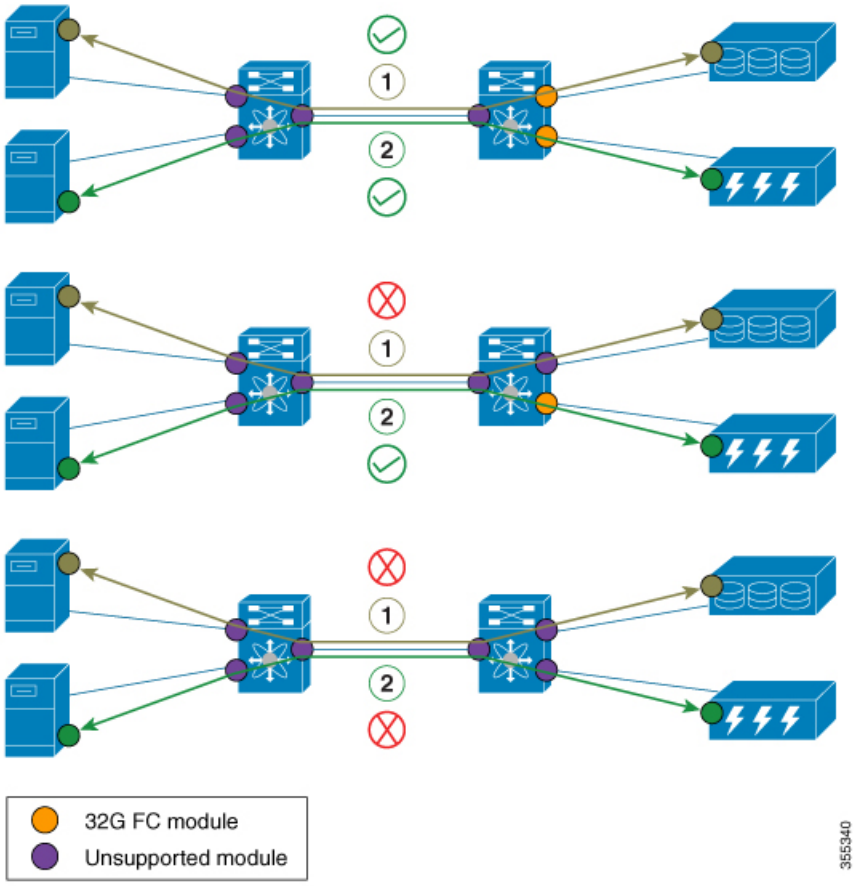

In the context of a topology, we recommend that you enable the flow metrics collection on the interfaces where all the flows are passing through the interface at least once.

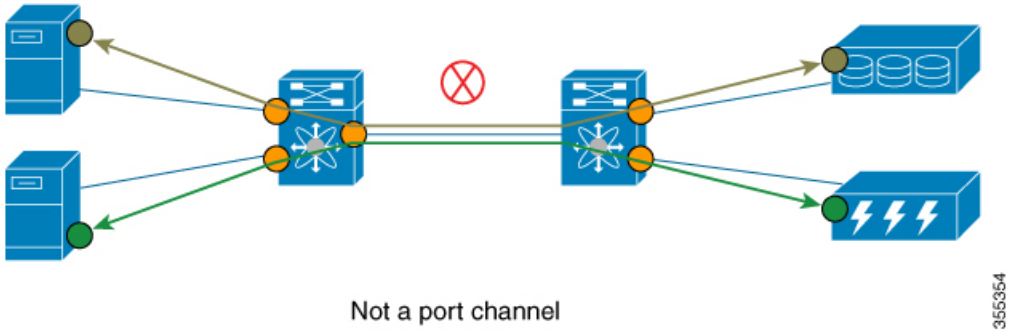
The following image shows the functionality of the SAN Analytics feature when supported and unsupported modules (16-Gbps Fibre Channel, Cisco MDS 9700 40-Gbps 24-Port FCoE Module (DS-X9824-960K9), Cisco MDS 24/10-Port SAN Extension Module (DS-X9334-K9), and so on) are used in SAN.

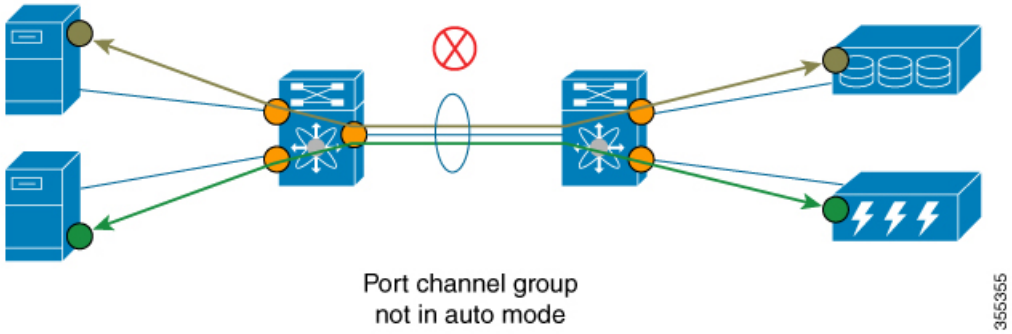
**Note** The numbers 1 and 2 in the Figure 13: Functionality of The SAN Analytics Feature When Supported and Unsupported Modules are Used represent flows.

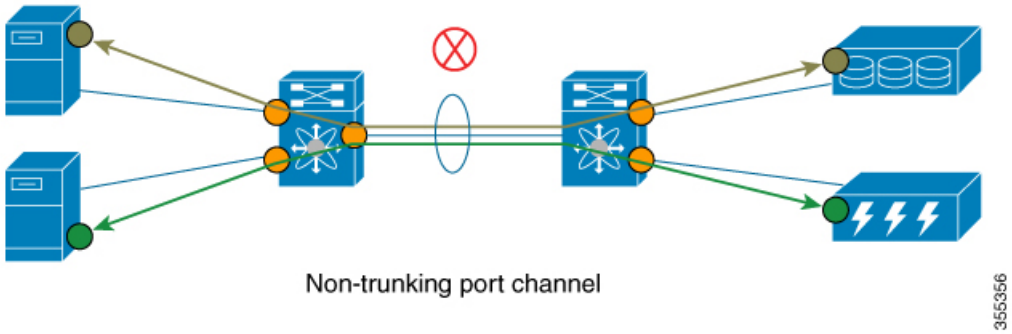
*Figure 13: Functionality of The SAN Analytics Feature When Supported and Unsupported Modules are Used*




1
2
1
2
1
2
32G FC module
Unsupported module
355340


Not a port channel
355354


Port channel group
not in auto mode
355355


Non-trunking port channel
355356

# OnBoard Querying

After you run a query, the flow metrics are collected, stored in a database, and displayed.

## Schema for Querying Metrics

Schema is used to display data, that is of interest to a user, that is stored in a database. Metrics are maintained in the database in the form of various view instance. These view instances can be retrieved using queries. For information on view types, see Views, on page 22.

## Query Syntax

The following is the query syntax that is used in the pull query, push query, clearing metrics, and purging views:

**select all** | *column1*[, *column2*, *column3*, …] **from** *analytics_type.view_type* [**where** *filter_list1* [**and** *filter_list2* …]] [**sort** *column*] [**limit** *number*]

The following are the elements of the query syntax:

- *analytics_type*—Specifies the analytics type. Only the *fc-scsi* type is supported in Cisco MDS NX-OS Release 8.2(1).
- *view_type*—Specifies the view type of the metric database. The syntax is used to run queries on it. The syntax supports only one query on a single view type. For a list of view types supported, see List of Supported View Types, on page 22.
- *column*—Specifies the flow metrics. A view instance contains multiple columns.
- *filter_list*—Specifies filters to extract only the view instances and metrics of a view instance. You can use the conditions on a flow metric column whose type is a key value or on a view instance column. You can also use the AND operator for filtering. For a list of view types supported, see List of Supported View Types, on page 22.
- **sort**—Specifies that the results in a column should be sorted. Sorting is performed before the limit operation is performed.
- **limit**—Specifies that the number of metrics returned in a result should be limited.

**Note** The sort, limit, and where options in the *"query_string"* can only be used on the *key* fields and the metrics are sorted only in ascending order. For more information on the *key* fields, see Flow Metrics.

## Query Semantic Rules

The following are the rules for constructing query semantics:

- The **select**, **from**, **where**, **sort**, and **limit** conditions should be used in the same order as described in Query Syntax, on page 21.

- The list of columns under the **select** condition should belong to the schema that corresponds to the *view_type* under the **from** condition.
- The **where** condition is allowed only on flow metric fields whose type is a key value. For information about the flow metric fields whose type is a key value, see List of Supported View Types, on page 22.
- The **sort** condition must be a *metric* field and should be present among the columns listed under the **select** condition.

# Views

A view is a representation of the flow metrics with respect to a port, initiator, target, LUN, or any valid combination of these. Each view type supports specific flow metrics. Long names in the flow metrics are used for OnBoard queries and short names are used for telemetry. For more information, see Flow Metrics.

## List of Supported View Types

The following table lists the supported view types:

***Table 4: Supported View Types***

| View Type | Description | Key |
|---|---|---|
| port | A port's table contains metadata and IO metrics for ports on a switch. | port |
| logical_port | A logical port table contains metadata and IO metrics for VSANs configured for ports on a switch. | port and vsan |
| app | An app table contains metadata and IO-metrics for the applications hosted behind various ports on switches that are performing IO operations. | port and app-id |
| scsi_target | A target table contains metadata and IO metrics for SCSI targets deployed behind various ports on a switch that executes IO operations. | port, vsan, and scsi-target-fc-id |
| scsi_initiator | An initiator table contains metadata and IO metrics for SCSI initiators deployed behind various ports on a switch that initiates IO operations. | port, vsan, and scsi-initiator-fc-id |
| scsi_target_app | A target app table contains metadata and IO metrics for the applications whose data are hosted on various targets. | port, vsan, scsi-target-fc-id, and app-id |

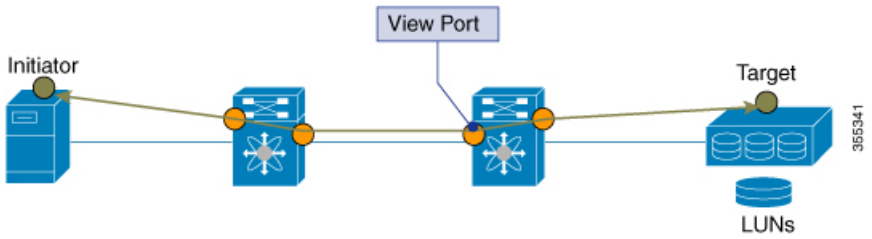
| View Type | Description | Key |
|---|---|---|
| scsi_initiator_app | An initiator app table contains metadata and IO metrics for the applications for which initiators initiate IO operations. | port, vsan, scsi-initiator-fc-id, and app-id |
| scs_target_it_flow | A target initiator-target (IT) flow table contains metadata and IO metrics for IT flows associated with various SCSI targets. | port, vsan, scsi-target-fc-id, and scsi-initiator-fc-id |
| scsi_initiator_it_flow | An initiator IT flow table contains metadata and IO metrics for the IT flows associated with various SCSI initiators. | port, vsan, scsi-initiator-fc-id, and scsi-target-fc-id |
| scsi_target_tl_flow | A target target-LUN (TL) flow table contains metadata and IO metrics for the LUNs associated with various SCSI targets. | port, vsan, scsi-target-fc-id, and lun-id |
| scsi_target_itl_flow | A target initiator-target-LUN (ITL) flow table contains metadata and IO metrics for ITL flows associated with various SCSI targets. | port, vsan, scsi-target-fc-id, scsi-initiator-fc-id, and lun-id |
| scsi_initiator_itl_flow | An initiator ITL flow table contains metadata and IO metrics for the ITL flows associated with various SCSI initiators. | port, vsan, scsi-initiator-fc-id, scsi-target-fc-id, and lun-id |
| scsi_target_io | A target IO table contains IO transaction details for the active IOs that various SCSI targets execute. | port, vsan, scsi-target-fc-id, scsi-initiator-fc-id, and ox-id |
| scsi_initiator_io | An initiator IO table records contain IO transaction details for the active IOs that various SCSI initiators initiate. | port, vsan, scsi-initiator-fc-id, scsi-target-fc-id, and ox-id |

## View Types Representation

The following images show graphical representation of some of the view types supported in the SAN Analytics feature.

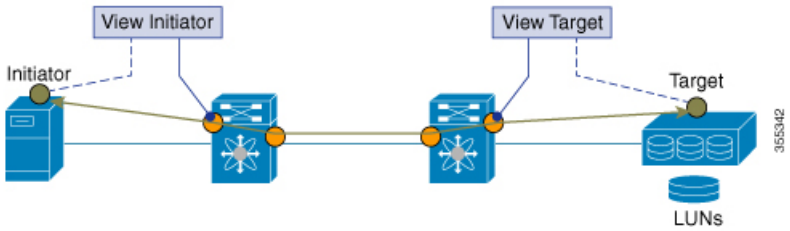
The following image shows the flow metrics as viewed from a port:

*Figure 14: Port View Type*



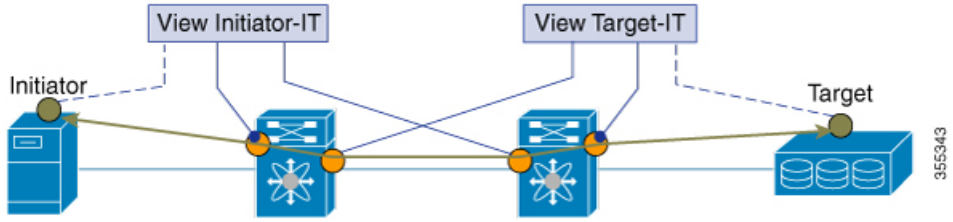

The following image shows the flow metrics as viewed from a port of an initiator and target:

*Figure 15: Initiator-Target View Type*



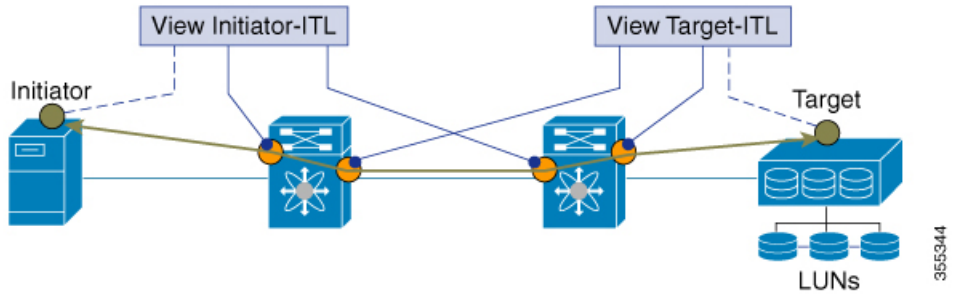

The following image shows the flow metrics as viewed from an initiator to a target and from a target to an initiator:

*Figure 16: Initiator-Target IT View Type*



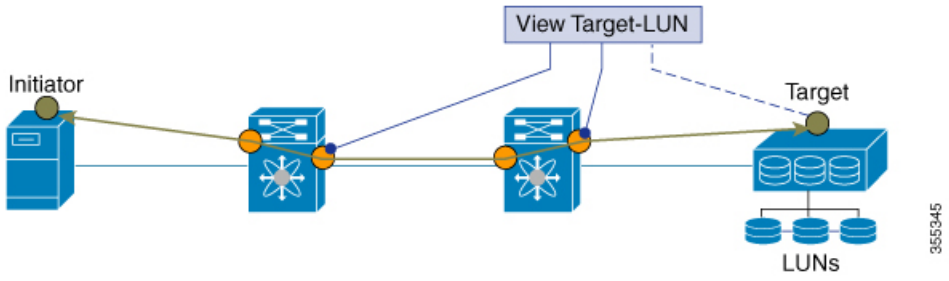

The following image shows the flow metrics as viewed from an initiator to a target and from a target to an initiator, with LUN metrics included:

*Figure 17: Initiator-Target ITL View Type*




The following image shows the flow metrics as viewed from a port on a target, with LUN metrics included:

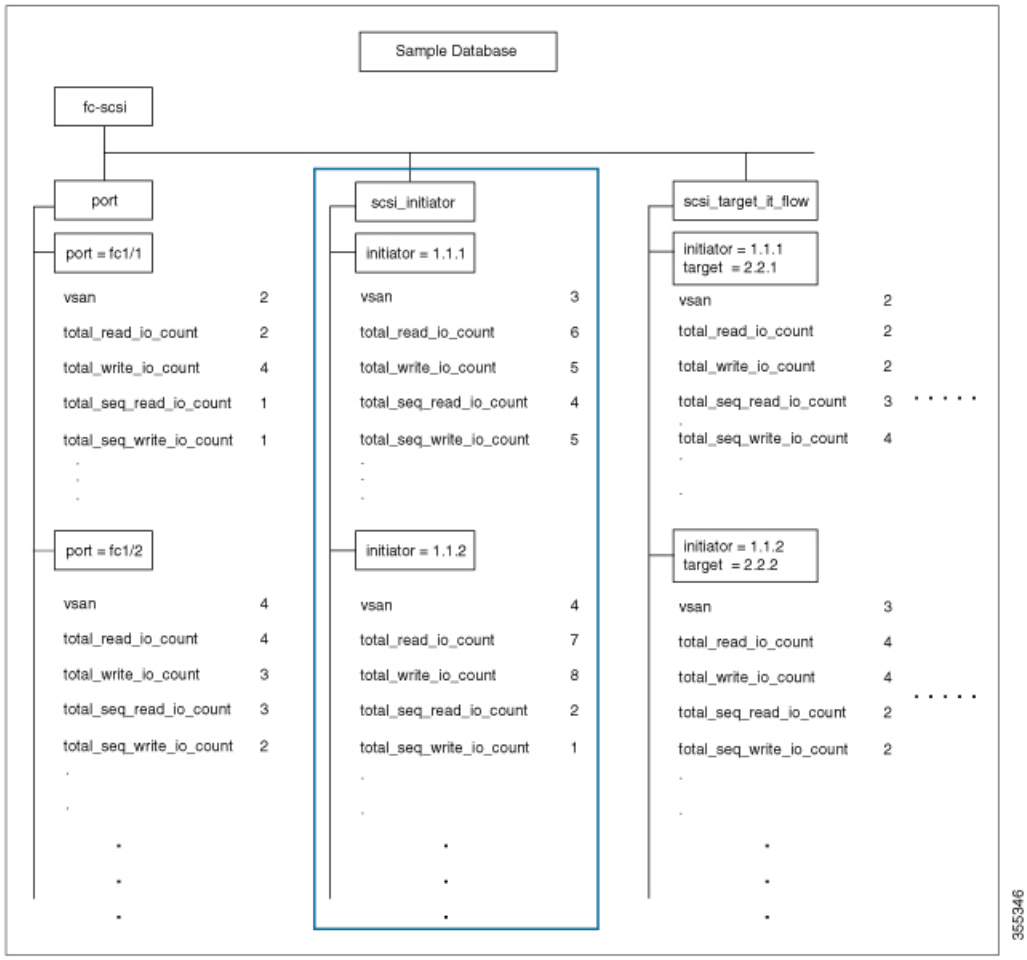
*Figure 18: Target-LUN View Type*




# Examples: Configuring Query Syntax

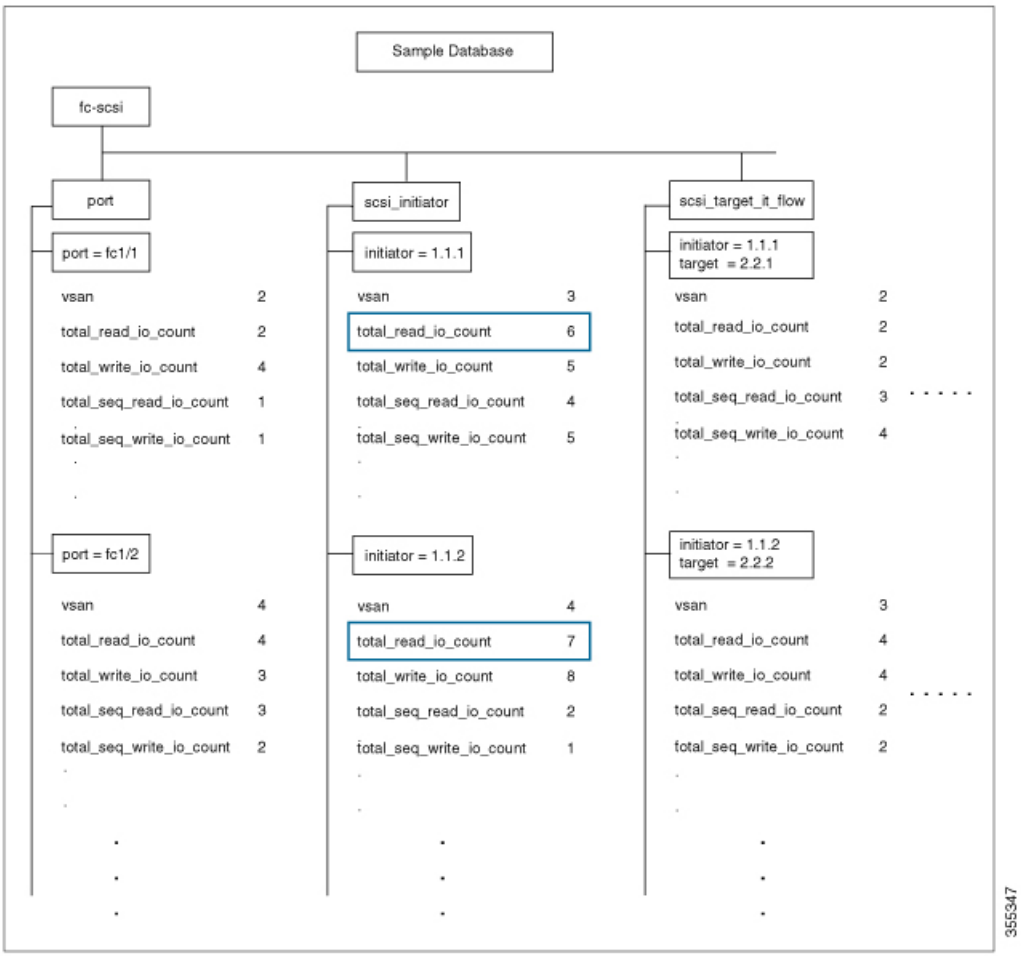
The **show analytics query 'select all from fc-scsi.scsi_initiator'** command provides an output of the flow metrics of all the initiators, as seen in the sample database shown in the following image:

*Figure 19: Flow Metrics of all the Initiators*



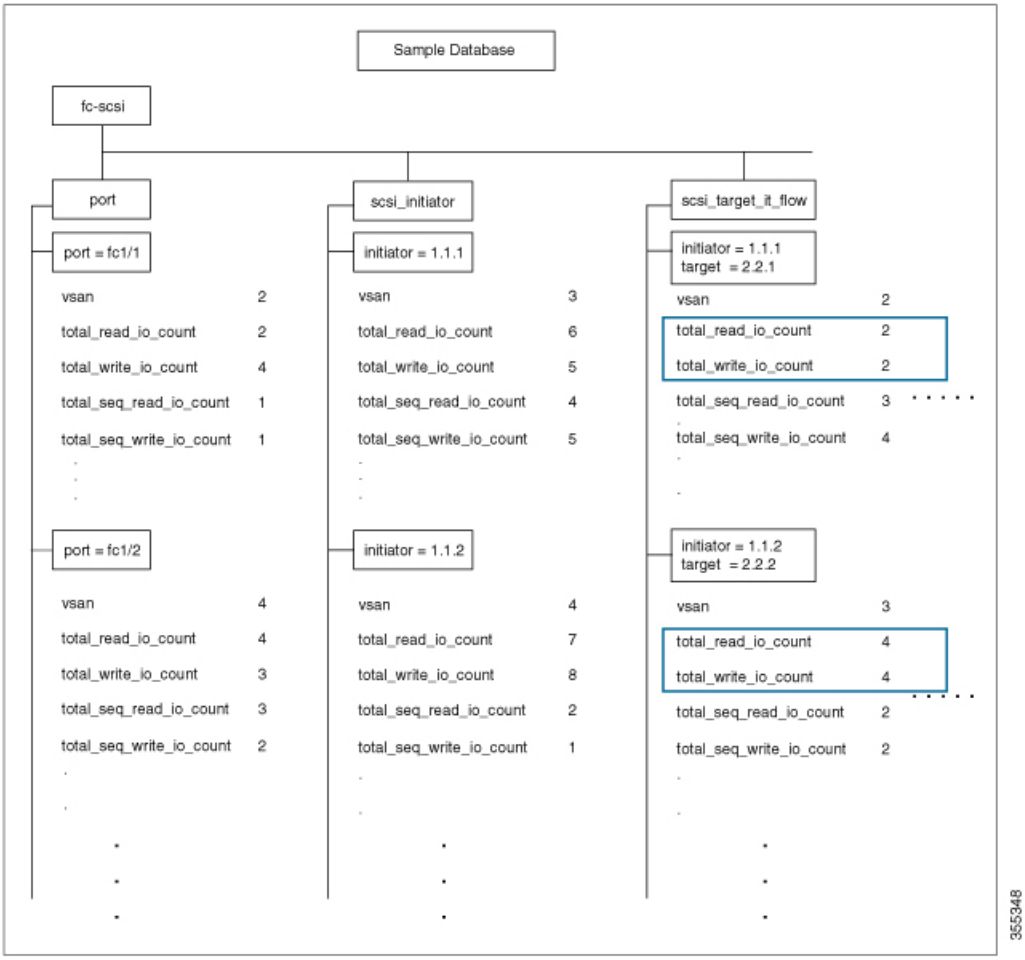
The **show analytics query 'select total_read_io_count from fc-scsi.scsi_initiator'** command provides an output of a target's total_read_io_count flow metrics, as seen in the sample database in the following image:

***Figure 20: Flow Metrics of a Target's Total Read IO Count***



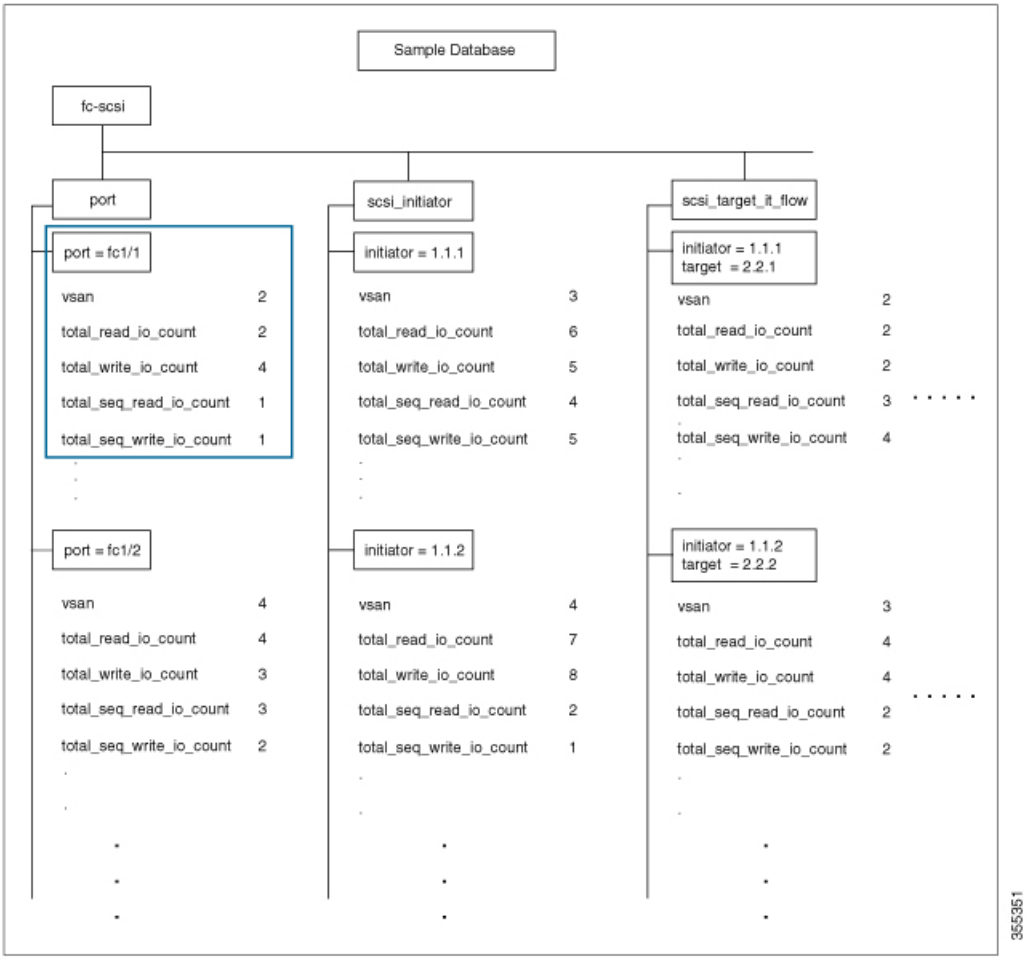

The **show analytics query 'select total_read_io_count,total_write_io_count from fc-scsi.scsi_target_it_flow'** command provides an output of an initiator's and a target's total_read_io_count and total_write_io_count flow metrics viewed from the target, as seen in the sample database in the following image:

*Figure 21: Flow Metrics of an Initiator's and Target's Total Read IO Count and Total Write IO Count*



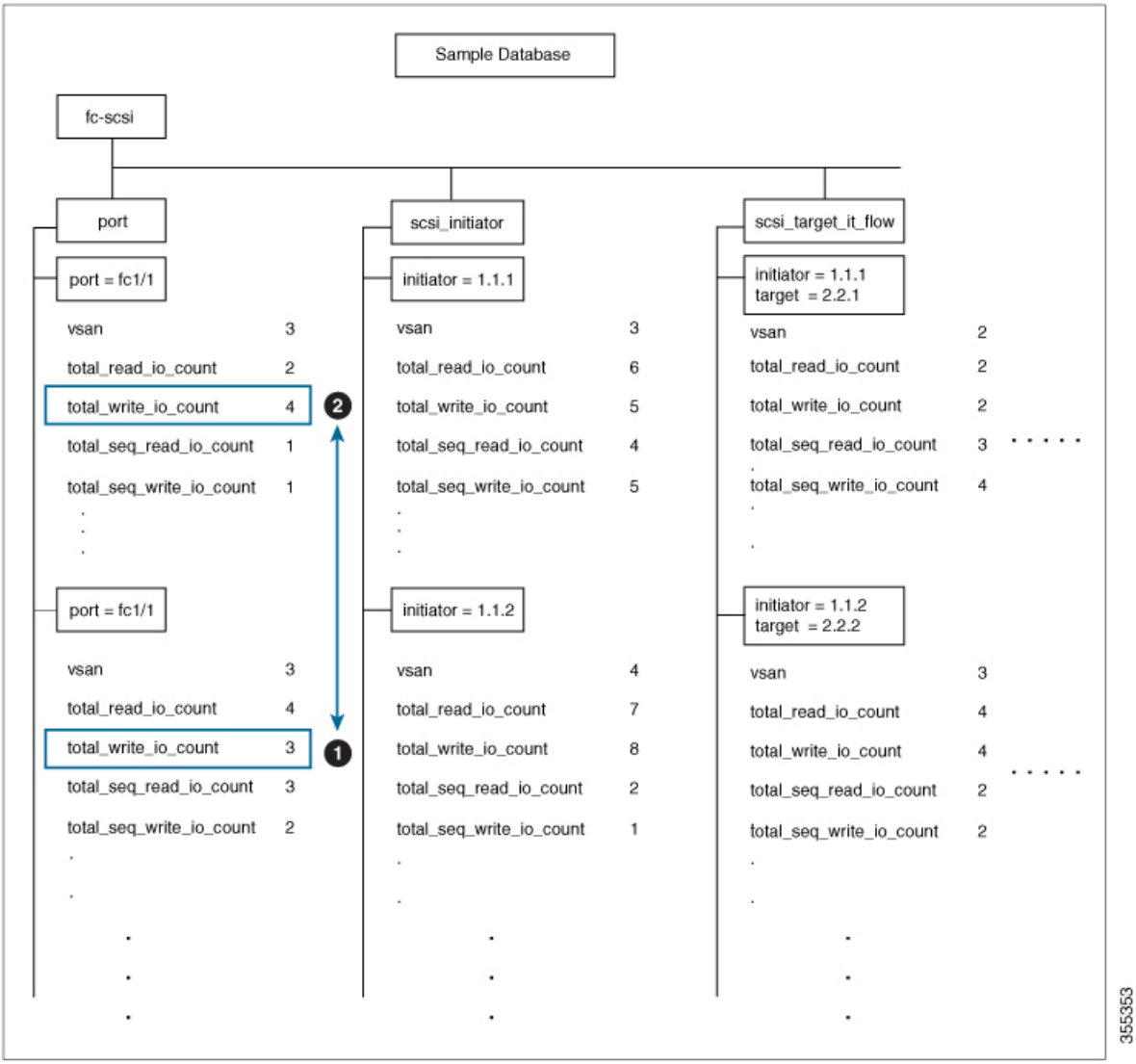

The **show analytics query 'select all from fc-scsi.port where port=fc1/1 and vsan=2 limit 1'** command provides an output of a port's flow metrics that are a part of port fc1/1, VSAN 2, with the number of records is limited to one, as seen in the sample database in the following image:

*Figure 22: Flow Metrics of the Port FC 1/1 That Belongs to VSAN 2 With the Number of Records Limited to One*




The **show analytics query 'select all from fc-scsi.scsi_initiator where port=fc1/1 and vsan=4 sort total_write_io_count'** command provides an output of an initiator's total_write_io_count flow metrics that are a part of port fc1/1 and VSAN 4, and the output is sorted, as seen in the sample database in the following image:

*Figure 23: Flow Metrics of an Initiator's Total Write IO Count That Belongs to Port FC1/1 and VSAN 4 With the Output Sorted*



# Configuring SAN Analytics

Enable the SAN Analytics feature on both a switch and its interfaces in order to enable flow metric collection from the interfaces.

**Note** To use the SAN Analytics feature, you must install an appropriate license package using the **install license** command. For more information, see the Cisco MDS 9000 Series Licensing Guide.

## Enabling SAN Analytics

**Note** The SAN Analytics feature is disabled by default.

To enable the SAN Analytics feature on a switch, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Enable the SAN Analytics feature on the switch:

switch(config)# **feature analytics**

## Disabling SAN Analytics

To disable the SAN Analytics feature on a switch, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Disable the SAN Analytics feature on the switch:

switch(config)# **no feature analytics**

## Enabling SAN Analytics on an Interface

To enable the SAN Analytics feature on an interface, perform these steps:

**Before you begin**

**Note** The SAN Analytics feature is disabled by default on all interfaces.

- Enable the SAN Analytics feature on the switch. See the Enabling SAN Analytics, on page 30 section.
- In port channels, enable the SAN Analytics feature on all the interfaces.

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Select a Fibre Channel interface or a range of interfaces and enter interface configuration submode:

switch(config)# **interface fc** *slot number/port number*

**Note** You can also specify the range for interfaces using the **interface fc** *slot number/port number - port number* , **fc** *slot number/port number - port number* command. The spaces are required before and after the dash ( - ) and before and after the comma ( , ).

**Step 3** Enable the SAN Analytics feature on the selected interface:

switch(config-if)# **analytics type fc-scsi**

**Note** Only the fc-scsi type is supported in Cisco MDS NX-OS Release 8.2(1) or later releases.

# Disabling SAN Analytics on an Interface

To disable the SAN Analytics feature on an interface, perform these steps:

**Before you begin**

In port channels, disable the SAN Analytics feature on all the interfaces.

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Select a Fibre Channel interface or a range of interfaces and enter interface configuration submode:

switch(config)# **interface fc** *slot number/port number*

**Note** You can also specify the range for interfaces using the **interface fc** *slot number/port number - port number* , **fc** *slot number/port number - port number* command. The spaces are required before and after the dash ( - ) and before and after the comma ( , ).

**Step 3** Disable the SAN Analytics feature on the selected interface:

switch(config-if)# **no analytics type fc-scsi**

# Enabling Port Sampling

The Port Sampling feature is useful when the NPU load is high and you cannot reduce the number of ports being monitored on a module. In such a situation, the load on the NPU can be reduced by sampling a subset of the monitored ports at a specified port sampling interval. Use the **show analytics port-sampling module** *number* command to check the NPU load.

**Note** Port sampling is disabled by default and continuous monitoring is enabled on all the analytics enabled ports. For more information on port sampling, see Port Sampling, on page 12.

To enable port sampling on a module, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Enable port sampling on a module:

switch# **analytics port-sampling module** *number* **size** *number* **interval** *seconds*

# Disabling Port Sampling

To disable port sampling on a module, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Disable port sampling on a module and go back to the default mode of monitoring all analytics enabled ports with the configured streaming interval:

switch# **no analytics port-sampling module** *number*

# Example: Configuring SAN Analytics

**Note** Port sampling is supported only in Cisco MDS NX-OS Release 8.3(1) or later releases.

This example shows how to enable the SAN Analytics feature on a switch:

```
switch# configure terminal
switch(config)# feature analytics
```

This example shows how to disable the SAN Analytics feature on a switch:

```
switch# configure terminal
switch(config)# no feature analytics
```

This example shows how to enable SAN Analytics on an interface:

```
switch# configure terminal
switch(config)# interface fc 1/1
switch(config-if)# analytics type fc-scsi
```

This example shows how to disable the SAN Analytics feature on an interface:

```
switch# configure terminal
switch(config)# interface fc 1/1
switch(config-if)# no analytics type fc-scsi
```

This example shows how to enable the SAN Analytics feature on a range of interfaces:

```
switch# configure terminal
switch(config)# interface fc1/1 - 4 , fc2/1 - 3
switch(config-if)# analytics type fc-scsi
```

This example shows how to disable the SAN Analytics feature on a range of interfaces:

```
switch# configure terminal
switch(config)# interface fc1/1 - 4 , fc2/1 - 3
switch(config-if)# no analytics type fc-scsi
```

This example shows how to enable port sampling on a module with port sampling interval of 35 seconds:

```
switch# configure terminal
switch(config)# analytics port-sampling module 2 size 12 interval 35
```

This example shows how to disable port sampling on a module and go back to the default mode of monitoring all analytics enabled ports with the configured streaming interval:

```
switch# configure terminal
switch(config)# no analytics port-sampling module 2
```

# Constructing and Using Queries

Flow metrics are analyzed by using a *query string* that is in the form of a query syntax.

## Displaying the Installed Push Queries

To display the installed push queries, run this command:

switch# **show analytics query** {**all** | **name** *query_name*}

## Displaying the Results of a Push Query

To display the results of a push query, run this command:

switch# **show analytics query name** *query_name* **result**

## Executing a Pull Query

To execute a pull query, run this command:

switch# **show analytics query** *"query_string"* [**clear**] [**differential**]

**Note** Use the *"query_string"* to specify query semantics, such as **select**, **table**, **limit**, and so on, for example, *"Select all from fc-scsi.port"*.

## Configure a Push Query

To configure a push query, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Specify a query string and a timer value for the flow metrics to be displayed at specific intervals:

switch(config)# **analytics query** *"query_string"* **name** *query_name* **type periodic** [**interval** *seconds*] [**clear**] [**differential**]

Only one push query using a *"query_string"* is allowed at a time. If you try to configure a duplicate push query name, a message is returned stating that the current configuration is a duplicate.

**Note** Pull query, push query, and overlay CLI are applicable only on interfaces where the SAN Analytics feature is enabled.

# Removing a Configured Push Query

To remove a configured push query, perform these steps:

**Step 1** Enter global configuration mode:

switch# **configure terminal**

**Step 2** Remove a configured push query:

switch(config)# **no analytics name** *query_name*

# Clearing Metrics

**Note**

- The *"query_string"* must have the format *"select all from <view-name>"*.
- You can clear the flow metrics without installing a push query.

To reset all the flow metrics for a view instance that match the query string, run this command:

switch# **clear analytics query** *"query_string"*

**Note** Clear resets all metrics of a view instance, whereas purge deletes specific view instance and its associated metrics momentarily. After clearing the database, the database will continue to collect the metrics for the specified *"query_string"*.

# Purging Views

To delete a specific view instance and its associated metrics, run this command:

switch# **purge analytics query** *"query_string"*

**Note**

- The *"query_string"* must have the format *"select all from <view-name>"*.
- You can clear the flow metrics without installing a push query.
- The where clause in the purge query can accept only the *port* key field.
- Purge deletes specific view instance and its associated metrics, whereas clear resets all the metrics of a view instance momentarily. After purging the database, the database will continue to collect the metrics for the specified *"query_string"*.

# Displaying the Results of a Configured Push Query

The flow metrics that are displayed using the **show analytics query name** *query_name* **result** command are the refreshed metrics at that time interval when this command was executed. For example, if a push query is configured to refresh at an interval of every 30 seconds, and the **show analytics query name** *query_name* **result** command is executed after 35 seconds, the push query displays the flow metrics that were refreshed when the time interval was 30 seconds.

To display the flow metrics of a configured push query, run this command:

switch# **show analytics query name** *query_name* **result**

# Example: Constructing and Using Queries

**Note**

- The number after *"values"* in the output indicates the corresponding record's number.
- New metrics were added in Cisco MDS NX-OS Release 8.3(1) because of which the query results may slightly differ between Cisco MDS NX-OS Release 8.3(1) or later releases and Cisco MDS NX-OS Release 8.2(1).

This example shows the output of all the flow metrics of the initiator ITL flow view instance:

```
switch#  show analytics query 'select all from fc-scsi.scsi_initiator_itl_flow'
{ "values": {
        "1": {
                "port": "fc1/1",
                "vsan": "10",
                "app_id": "255",
                "initiator_id": "0xe80041",
                "target_id": "0xd60200",
                "lun": "0000-0000-0000-0000",
                "active_io_read_count": "0",
                "active_io_write_count": "1",
                "total_read_io_count": "0",
                "total_write_io_count": "1162370362",
                "total_seq_read_io_count": "0",
                "total_seq_write_io_count": "1",
                "total_read_io_time": "0",
                "total_write_io_time": "116204704658",
                "total_read_io_initiation_time": "0",
                "total_write_io_initiation_time": "43996934029",
                "total_read_io_bytes": "0",
                "total_write_io_bytes": "595133625344",
                "total_read_io_inter_gap_time": "0",
                "total_write_io_inter_gap_time": "41139462314556",
                "total_time_metric_based_read_io_count": "0",
                "total_time_metric_based_write_io_count": "1162370358",
                "total_time_metric_based_read_io_bytes": "0",
                "total_time_metric_based_write_io_bytes": "595133623296",
                "read_io_rate": "0",
                "peak_read_io_rate": "0",
                "write_io_rate": "7250",
                "peak_write_io_rate": "7304",
                "read_io_bandwidth": "0",
                "peak_read_io_bandwidth": "0",
```

```
            "write_io_bandwidth": "3712384",
            "peak_write_io_bandwidth": "3739904",
            "read_io_size_min": "0",
            "read_io_size_max": "0",
            "write_io_size_min": "512",
            "write_io_size_max": "512",
            "read_io_completion_time_min": "0",
            "read_io_completion_time_max": "0",
            "write_io_completion_time_min": "89",
            "write_io_completion_time_max": "416",
            "read_io_initiation_time_min": "0",
            "read_io_initiation_time_max": "0",
            "write_io_initiation_time_min": "34",
            "write_io_initiation_time_max": "116",
            "read_io_inter_gap_time_min": "0",
            "read_io_inter_gap_time_max": "0",
            "write_io_inter_gap_time_min": "31400",
            "write_io_inter_gap_time_max": "118222",
            "peak_active_io_read_count": "0",
            "peak_active_io_write_count": "5",
            "read_io_aborts": "0",
            "write_io_aborts": "0",
            "read_io_failures": "0",
            "write_io_failures": "0",
            "read_io_timeouts": "0",
            "write_io_timeouts": "1",
            "read_io_scsi_check_condition_count": "0",
            "write_io_scsi_check_condition_count": "0",
            "read_io_scsi_busy_count": "0",
            "write_io_scsi_busy_count": "0",
            "read_io_scsi_reservation_conflict_count": "0",
            "write_io_scsi_reservation_conflict_count": "0",
            "read_io_scsi_queue_full_count": "0",
            "write_io_scsi_queue_full_count": "0",
            "sampling_start_time": "1528535447",
            "sampling_end_time": "1528697457"
    },
.
.
.
    "5": {
            "port": "fc1/8",
            "vsan": "10",
            "app_id": "255",
            "initiator_id": "0xe80001",
            "target_id": "0xe800a1",
            "lun": "0000-0000-0000-0000",
            "active_io_read_count": "0",
            "active_io_write_count": "1",
            "total_read_io_count": "0",
            "total_write_io_count": "1138738309",
            "total_seq_read_io_count": "0",
            "total_seq_write_io_count": "1",
            "total_read_io_time": "0",
            "total_write_io_time": "109792480881",
            "total_read_io_initiation_time": "0",
            "total_write_io_initiation_time": "39239145641",
            "total_read_io_bytes": "0",
            "total_write_io_bytes": "583034014208",
            "total_read_io_inter_gap_time": "0",
            "total_write_io_inter_gap_time": "41479779998852",
            "total_time_metric_based_read_io_count": "0",
            "total_time_metric_based_write_io_count": "1138738307",
            "total_time_metric_based_read_io_bytes": "0",
```

```
                "total_time_metric_based_write_io_bytes": "583034013184",
                "read_io_rate": "0",
                "peak_read_io_rate": "0",
                "write_io_rate": "7074",
                "peak_write_io_rate": "7903",
                "read_io_bandwidth": "0",
                "peak_read_io_bandwidth": "0",
                "write_io_bandwidth": "3622144",
                "peak_write_io_bandwidth": "4046336",
                "read_io_size_min": "0",
                "read_io_size_max": "0",
                "write_io_size_min": "512",
                "write_io_size_max": "512",
                "read_io_completion_time_min": "0",
                "read_io_completion_time_max": "0",
                "write_io_completion_time_min": "71",
                "write_io_completion_time_max": "3352",
                "read_io_initiation_time_min": "0",
                "read_io_initiation_time_max": "0",
                "write_io_initiation_time_min": "26",
                "write_io_initiation_time_max": "2427",
                "read_io_inter_gap_time_min": "0",
                "read_io_inter_gap_time_max": "0",
                "write_io_inter_gap_time_min": "25988",
                "write_io_inter_gap_time_max": "868452",
                "peak_active_io_read_count": "0",
                "peak_active_io_write_count": "5",
                "read_io_aborts": "0",
                "write_io_aborts": "0",
                "read_io_failures": "0",
                "write_io_failures": "0",
                "read_io_timeouts": "0",
                "write_io_timeouts": "1",
                "read_io_scsi_check_condition_count": "0",
                "write_io_scsi_check_condition_count": "0",
                "read_io_scsi_busy_count": "0",
                "write_io_scsi_busy_count": "0",
                "read_io_scsi_reservation_conflict_count": "0",
                "write_io_scsi_reservation_conflict_count": "0",
                "read_io_scsi_queue_full_count": "0",
                "write_io_scsi_queue_full_count": "0",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697457"
        }
}}
```

This example shows the output of requesting specific flow metrics for a specific initiator ID of an initiator ITL flow view instance:

```
switch#   show analytics query 'select
port,initiator_id,target_id,lun,total_read_io_count,total_write_io_count,read_io_rate,write_io_rate
 from fc-scsi.scsi_initiator_itl_flow where initiator_id=0xe80001'
{ "values": {
        "1": {
                "port": "fc1/8",
                "initiator_id": "0xe80001",
                "target_id": "0xe800a1",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1139010960",
                "read_io_rate": "0",
                "write_io_rate": "7071",
```

```
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697495"
        }
}}
```

This example shows the output of requesting specific flow metrics for a specific initiator ID and LUN of an initiator ITL flow view instance:

```
switch#   show analytics query 'select
port,initiator_id,target_id,lun,total_read_io_count,total_write_io_count,read_io_rate,write_io_rate
 from fc-scsi.scsi_initiator_itl_flow where initiator_id=0xe80001 and lun=0000-0000-0000-0000'
{ "values": {
        "1": {
                "port": "fc1/8",
                "initiator_id": "0xe80001",
                "target_id": "0xe800a1",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1139453979",
                "read_io_rate": "0",
                "write_io_rate": "7070",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697559"
        }
}}
```

This example shows the output of specific flow metrics for a specific LUN, with the output sorted for the write_io_rate metrics of a target ITL flow view instance:

```
switch#   show analytics query 'select
port,initiator_id,target_id,lun,total_read_io_count,total_write_io_count,read_io_rate,write_io_rate
 from fc-scsi.scsi_target_itl_flow where lun=0000-0000-0000-0000 sort write_io_rate'
{ "values": {
        "1": {
                "port": "fc1/6",
                "initiator_id": "0xe80020",
                "target_id": "0xd60040",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1103394068",
                "read_io_rate": "0",
                "write_io_rate": "6882",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697630"
        },
        "2": {
                "port": "fc1/6",
                "initiator_id": "0xe80021",
                "target_id": "0xe80056",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1119199742",
                "read_io_rate": "0",
                "write_io_rate": "6946",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697630"
        },
        "3": {
                "port": "fc1/8",
                "initiator_id": "0xe80000",
                "target_id": "0xe80042",
```

```
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1119506589",
                "read_io_rate": "0",
                "write_io_rate": "6948",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697630"
        },
        "4": {
                "port": "fc1/8",
                "initiator_id": "0xe80001",
                "target_id": "0xe800a1",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1139953183",
                "read_io_rate": "0",
                "write_io_rate": "7068",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697630"
        },
        "5": {
                "port": "fc1/1",
                "initiator_id": "0xe80041",
                "target_id": "0xd60200",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1163615698",
                "read_io_rate": "0",
                "write_io_rate": "7247",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697630"
        }
}}
```

This example shows the output of specific flow metrics for a specific LUN, with the output limited to three records and sorted for the write_io_rate metrics of an initiator ITL flow view instance:

```
switch#   show analytics query 'select
port,initiator_id,target_id,lun,total_read_io_count,total_write_io_count,read_io_rate,write_io_rate
 from fc-scsi.scsi_initiator_itl_flow where lun=0000-0000-0000-0000 sort write_io_rate limit
 3'
{ "values": {
        "1": {
                "port": "fc1/6",
                "initiator_id": "0xe80020",
                "target_id": "0xd60040",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1103901828",
                "read_io_rate": "0",
                "write_io_rate": "6885",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697704"
        },
        "2": {
                "port": "fc1/8",
                "initiator_id": "0xe80000",
                "target_id": "0xe80042",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1120018575",
                "read_io_rate": "0",
```

```
                "write_io_rate": "6940",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697704"
        },
        "3": {
                "port": "fc1/6",
                "initiator_id": "0xe80021",
                "target_id": "0xe80056",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1119711583",
                "read_io_rate": "0",
                "write_io_rate": "6942",
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697704"
        }
}}
```

This example shows the output of specific flow metrics for a specific LUN and target ID of an initiator ITL flow view instance:

```
switch#   show analytics query 'select
port,initiator_id,target_id,lun,total_read_io_count,total_write_io_count,read_io_rate,write_io_rate
 from fc-scsi.scsi_initiator_itl_flow where lun=0000-0000-0000-0000 and target_id=0xe800a1'
{ "values": {
        "1": {
                "port": "fc1/8",
                "initiator_id": "0xe80001",
                "target_id": "0xe800a1",
                "lun": "0000-0000-0000-0000",
                "total_read_io_count": "0",
                "total_write_io_count": "1139010960",
                "read_io_rate": "0",
                "write_io_rate": "7071"
                "sampling_start_time": "1528535447",
                "sampling_end_time": "1528697495"
        }
}}
```

This example shows how to configure a push query when the duration to refresh the flow metrics is set to the default duration of 30 seconds:

```
switch# configure terminal
switch(config)# analytics query 'select all from fc-scsi.scsi_initiator_itl_flow' name
initiator_itl_flow type periodic
switch(config)# show analytics query name initiator_itl_flow result
{ "values": {
        "1": {
                "port": "fc1/1",
                "vsan": "10",
                "app_id": "255",
                "initiator_id": "0xe80041",
                "target_id": "0xd60200",
                "lun": "0000-0000-0000-0000",
                "active_io_read_count": "0",
                "active_io_write_count": "1",
                "total_read_io_count": "0",
                "total_write_io_count": "1162370362",
                "total_seq_read_io_count": "0",
                "total_seq_write_io_count": "1",
                "total_read_io_time": "0",
```

```
            "total_write_io_time": "116204704658",
            "total_read_io_initiation_time": "0",
            "total_write_io_initiation_time": "43996934029",
            "total_read_io_bytes": "0",
            "total_write_io_bytes": "595133625344",
            "total_read_io_inter_gap_time": "0",
            "total_write_io_inter_gap_time": "41139462314556",
            "total_time_metric_based_read_io_count": "0",
            "total_time_metric_based_write_io_count": "1162370358",
            "total_time_metric_based_read_io_bytes": "0",
            "total_time_metric_based_write_io_bytes": "595133623296",
            "read_io_rate": "0",
            "peak_read_io_rate": "0",
            "write_io_rate": "7250",
            "peak_write_io_rate": "7304",
            "read_io_bandwidth": "0",
            "peak_read_io_bandwidth": "0",
            "write_io_bandwidth": "3712384",
            "peak_write_io_bandwidth": "3739904",
            "read_io_size_min": "0",
            "read_io_size_max": "0",
            "write_io_size_min": "512",
            "write_io_size_max": "512",
            "read_io_completion_time_min": "0",
            "read_io_completion_time_max": "0",
            "write_io_completion_time_min": "89",
            "write_io_completion_time_max": "416",
            "read_io_initiation_time_min": "0",
            "read_io_initiation_time_max": "0",
            "write_io_initiation_time_min": "34",
            "write_io_initiation_time_max": "116",
            "read_io_inter_gap_time_min": "0",
            "read_io_inter_gap_time_max": "0",
            "write_io_inter_gap_time_min": "31400",
            "write_io_inter_gap_time_max": "118222",
            "peak_active_io_read_count": "0",
            "peak_active_io_write_count": "5",
            "read_io_aborts": "0",
            "write_io_aborts": "0",
            "read_io_failures": "0",
            "write_io_failures": "0",
            "read_io_timeouts": "0",
            "write_io_timeouts": "1",
            "read_io_scsi_check_condition_count": "0",
            "write_io_scsi_check_condition_count": "0",
            "read_io_scsi_busy_count": "0",
            "write_io_scsi_busy_count": "0",
            "read_io_scsi_reservation_conflict_count": "0",
            "write_io_scsi_reservation_conflict_count": "0",
            "read_io_scsi_queue_full_count": "0",
            "write_io_scsi_queue_full_count": "0",
            "sampling_start_time": "1528535447",
            "sampling_end_time": "1528697457"
    },
    .
    .
    .
    "5": {
            "port": "fc1/8",
            "vsan": "10",
            "app_id": "255",
            "initiator_id": "0xe80001",
            "target_id": "0xe800a1",
            "lun": "0000-0000-0000-0000",
```

```
                    "active_io_read_count": "0",
                    "active_io_write_count": "1",
                    "total_read_io_count": "0",
                    "total_write_io_count": "1138738309",
                    "total_seq_read_io_count": "0",
                    "total_seq_write_io_count": "1",
                    "total_read_io_time": "0",
                    "total_write_io_time": "109792480881",
                    "total_read_io_initiation_time": "0",
                    "total_write_io_initiation_time": "39239145641",
                    "total_read_io_bytes": "0",
                    "total_write_io_bytes": "583034014208",
                    "total_read_io_inter_gap_time": "0",
                    "total_write_io_inter_gap_time": "41479779998852",
                    "total_time_metric_based_read_io_count": "0",
                    "total_time_metric_based_write_io_count": "1138738307",
                    "total_time_metric_based_read_io_bytes": "0",
                    "total_time_metric_based_write_io_bytes": "583034013184",
                    "read_io_rate": "0",
                    "peak_read_io_rate": "0",
                    "write_io_rate": "7074",
                    "peak_write_io_rate": "7903",
                    "read_io_bandwidth": "0",
                    "peak_read_io_bandwidth": "0",
                    "write_io_bandwidth": "3622144",
                    "peak_write_io_bandwidth": "4046336",
                    "read_io_size_min": "0",
                    "read_io_size_max": "0",
                    "write_io_size_min": "512",
                    "write_io_size_max": "512",
                    "read_io_completion_time_min": "0",
                    "read_io_completion_time_max": "0",
                    "write_io_completion_time_min": "71",
                    "write_io_completion_time_max": "3352",
                    "read_io_initiation_time_min": "0",
                    "read_io_initiation_time_max": "0",
                    "write_io_initiation_time_min": "26",
                    "write_io_initiation_time_max": "2427",
                    "read_io_inter_gap_time_min": "0",
                    "read_io_inter_gap_time_max": "0",
                    "write_io_inter_gap_time_min": "25988",
                    "write_io_inter_gap_time_max": "868452",
                    "peak_active_io_read_count": "0",
                    "peak_active_io_write_count": "5",
                    "read_io_aborts": "0",
                    "write_io_aborts": "0",
                    "read_io_failures": "0",
                    "write_io_failures": "0",
                    "read_io_timeouts": "0",
                    "write_io_timeouts": "1",
                    "read_io_scsi_check_condition_count": "0",
                    "write_io_scsi_check_condition_count": "0",
                    "read_io_scsi_busy_count": "0",
                    "write_io_scsi_busy_count": "0",
                    "read_io_scsi_reservation_conflict_count": "0",
                    "write_io_scsi_reservation_conflict_count": "0",
                    "read_io_scsi_queue_full_count": "0",
                    "write_io_scsi_queue_full_count": "0",
                    "sampling_start_time": "1528535447",
                    "sampling_end_time": "1528697457"
                }
        }}
```

This example shows how to clear all the minimum, maximum, and peak flow metrics:

- This example show the output before clearing the all the minimum, maximum, and peak flow metrics:

**Note** You must execute the clear command twice for the first time for clearing all the minimum, maximum, and peak flow metrics. Thereafter, you can execute the clear command once for clearing the flow metrics.

```
switch# show analytics query "select all from  fc-scsi.scsi_target_itl_flow where
port=fc1/17" clear
{ "values": {
        "1": {
                "port": "fc1/17",
                "vsan": "1",
                "app_id": "255",
                "target_id": "0xef0040",
                "initiator_id": "0xef0000",
                "lun": "0000-0000-0000-0000",
                "active_io_read_count": "0",
                "active_io_write_count": "1",
                "total_read_io_count": "0",
                "total_write_io_count": "84701",
                "total_seq_read_io_count": "0",
                "total_seq_write_io_count": "1",
                "total_read_io_time": "0",
                "total_write_io_time": "7007132",
                "total_read_io_initiation_time": "0",
                "total_write_io_initiation_time": "2421756",
                "total_read_io_bytes": "0",
                "total_write_io_bytes": "86733824",
                "total_read_io_inter_gap_time": "0",
                "total_write_io_inter_gap_time": "2508109021",
                "total_time_metric_based_read_io_count": "0",
                "total_time_metric_based_write_io_count": "84701",
                "total_time_metric_based_read_io_bytes": "0",
                "total_time_metric_based_write_io_bytes": "86733824",
                "read_io_rate": "0",
                "peak_read_io_rate": "0",
                "write_io_rate": "8711",
                "peak_write_io_rate": "8711",
                "read_io_bandwidth": "0",
                "peak_read_io_bandwidth": "0",
                "write_io_bandwidth": "8920576",
                "peak_write_io_bandwidth": "8920576",
                "read_io_size_min": "0",
                "read_io_size_max": "0",
                "write_io_size_min": "1024",
                "write_io_size_max": "1024",
                "read_io_completion_time_min": "0",
                "read_io_completion_time_max": "0",
                "write_io_completion_time_min": "74",
                "write_io_completion_time_max": "844",
                "read_io_initiation_time_min": "0",
                "read_io_initiation_time_max": "0",
                "write_io_initiation_time_min": "24",
                "write_io_initiation_time_max": "775",
                "read_io_inter_gap_time_min": "0",
                "read_io_inter_gap_time_max": "0",
                "write_io_inter_gap_time_min": "26903",
                "write_io_inter_gap_time_max": "287888",
```

```
                "peak_active_io_read_count": "0",
                "peak_active_io_write_count": "3",
                "read_io_aborts": "0",
                "write_io_aborts": "0",
                "read_io_failures": "0",
                "write_io_failures": "0",
                "read_io_timeouts": "0",
                "write_io_timeouts": "0",
                "read_io_scsi_check_condition_count": "0",
                "write_io_scsi_check_condition_count": "0",
                "read_io_scsi_busy_count": "0",
                "write_io_scsi_busy_count": "0",
                "read_io_scsi_reservation_conflict_count": "0",
                "write_io_scsi_reservation_conflict_count": "0",
                "read_io_scsi_queue_full_count": "0",
                "write_io_scsi_queue_full_count": "0",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530684301"
        },
}}
```

- This examples shows the output after clearing all the minimum, maximum, and peak flow metrics. The metrics that were cleared are highlighted in the output.

```
switch# show analytics query "select all from  fc-scsi.scsi_target_itl_flow where
port=fc1/17" clear
{ "values": {
        "1": {
                "port": "fc1/17",
               "vsan": "1",
                "app_id": "255",
                "target_id": "0xef0040",
                "initiator_id": "0xef0000",
                "lun": "0000-0000-0000-0000",
                "active_io_read_count": "0",
                "active_io_write_count": "0",
                "total_read_io_count": "0",
                "total_write_io_count": "800615",
                "total_seq_read_io_count": "0",
                "total_seq_write_io_count": "1",
                "total_read_io_time": "0",
                "total_write_io_time": "66090290",
                "total_read_io_initiation_time": "0",
                "total_write_io_initiation_time": "22793874",
                "total_read_io_bytes": "0",
                "total_write_io_bytes": "819829760",
                "total_read_io_inter_gap_time": "0",
                "total_write_io_inter_gap_time": "23702347887",
                "total_time_metric_based_read_io_count": "0",
                "total_time_metric_based_write_io_count": "800615",
                "total_time_metric_based_read_io_bytes": "0",
                "total_time_metric_based_write_io_bytes": "819829760",
                "read_io_rate": "0",
                "peak_read_io_rate": "0",
                "write_io_rate": "0",
                "peak_write_io_rate": "0",
                "read_io_bandwidth": "0",
                "peak_read_io_bandwidth": "0",
                "write_io_bandwidth": "0",
                "peak_write_io_bandwidth": "0",
                "read_io_size_min": "0",
                "read_io_size_max": "0",
```

```
                "write_io_size_min": "0",
                "write_io_size_max": "0",
                "read_io_completion_time_min": "0",
                "read_io_completion_time_max": "0",
                "write_io_completion_time_min": "0",
                "write_io_completion_time_max": "0",
                "read_io_initiation_time_min": "0",
                "read_io_initiation_time_max": "0",
                "write_io_initiation_time_min": "0",
                "write_io_initiation_time_max": "0",
                "read_io_inter_gap_time_min": "0",
                "read_io_inter_gap_time_max": "0",
                "write_io_inter_gap_time_min": "0",
                "write_io_inter_gap_time_max": "0",
                "peak_active_io_read_count": "0",
                "peak_active_io_write_count": "0",
                "read_io_aborts": "0",
                "write_io_aborts": "0",
                "read_io_failures": "0",
                "write_io_failures": "0",
                "read_io_timeouts": "0",
                "write_io_timeouts": "0",
                "read_io_scsi_check_condition_count": "0",
                "write_io_scsi_check_condition_count": "0",
                "read_io_scsi_busy_count": "0",
                "write_io_scsi_busy_count": "0",
                "read_io_scsi_reservation_conflict_count": "0",
                "write_io_scsi_reservation_conflict_count": "0",
                "read_io_scsi_queue_full_count": "0",
                "write_io_scsi_queue_full_count": "0",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530684428"
        },
}}
```

These examples show how to stream only the ITL flow metrics that have changed between streaming intervals:

- This example shows the output before using the differential option:

```
switch# show analytics query "select port, target_id,
initiator_id,lun,total_write_io_count from fc-scsi.scsi_target_itl_flow where port=fc1/17"
 differential
{ "values": {
        "1": {
                "port": "fc1/17",
                "target_id": "0xef0040",
               "initiator_id": "0xef0000",
                "lun": "0001-0000-0000-0000",
                "total_write_io_count": "1515601",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683484"
        },
        "2": {
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0020",
                "lun": "0000-0000-0000-0000",
                "total_write_io_count": "1515601",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683484"
        },
        "3": {
```

```
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0020",
                "lun": "0001-0000-0000-0000",
                "total_write_io_count": "1515600",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683484"
        },
        "4": {
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0000",
                "lun": "0000-0000-0000-0000",
                "total_write_io_count": "1515600",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683484"
        }
}}
```

- This example shows the output with the differential option and shows only the records that have changed:

```
switch# show analytics query "select port, target_id,
initiator_id,lun,total_write_io_count from fc-scsi.scsi_target_itl_flow where port=fc1/17"
 differential
{ "values": {
       "1": {
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0000",
                "lun": "0001-0000-0000-0000",
                "total_write_io_count": "1892021",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683534"
        },
        "2": {
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0020",
                "lun": "0000-0000-0000-0000",
                "total_write_io_count": "1892021",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683534"
        },
        "3": {
                "port": "fc1/17",
                "target_id": "0xef0040",
                "initiator_id": "0xef0000",
                "lun": "0000-0000-0000-0000",
                "total_write_io_count": "1892021",
                "sampling_start_time": "1530683133",
                "sampling_end_time": "1530683534"
        }
}}
```

This example shows how to remove an installed query name:

```
switch(config)# no analytics name initiator_itl_flow
```

These examples show how to clear flow metrics:

1. This example show the output before clearing flow metrics:

```
switch# show analytics query "select port,target_id,total_write_io_count,
total_write_io_bytes,total_time_metric_based_write_io_count,write_io_rate,
peak_write_io_rate,write_io_bandwidth,peak_write_io_bandwidth,
write_io_size_min,write_io_size_max,write_io_completion_time_min,
write_io_completion_time_max,write_io_initiation_time_min,
write_io_initiation_time_max,write_io_inter_gap_time_min,write_io_inter_gap_time_max
from fc-scsi.scsi_target where
target_id=0x650060"
{ "values": {
        "1": {
                "port": "fc3/17",
                "target_id": "0x650060",
                "total_write_io_count": "67350021",
                "total_write_io_bytes": "17655403905024",
                "total_time_metric_based_write_io_count": "67349761",
                "write_io_rate": "0",
                "peak_write_io_rate": "6300",
                "write_io_bandwidth": "0",
                "peak_write_io_bandwidth": "1651572736",
                "write_io_size_min": "262144",
                "write_io_size_max": "262144",
               "write_io_completion_time_min": "192",
                "write_io_completion_time_max": "9434",
                "write_io_initiation_time_min": "21",
                "write_io_initiation_time_max": "199",
                "write_io_inter_gap_time_min": "2553",
                "write_io_inter_gap_time_max": "358500",
                "sampling_start_time": "1531204359",
                "sampling_end_time": "1531215327"
        }
```

2. This example shows how to clear flow metrics:

**Note** Clearing metrics is allowed only on view instances and not on individual flow metrics.

```
switch# clear analytics query "select all from fc-scsi.scsi_target where
target_id=0x650060"
```

3. This example shows the output after clearing flow metrics:

```
switch# show analytics query "select port,target_id,total_write_io_count,
total_write_io_bytes,total_time_metric_based_write_io_count,write_io_rate,
peak_write_io_rate,write_io_bandwidth,peak_write_io_bandwidth,
write_io_size_min,write_io_size_max,write_io_completion_time_min,
write_io_completion_time_max,write_io_initiation_time_min,
write_io_initiation_time_max,write_io_inter_gap_time_min,write_io_inter_gap_time_max
from fc-scsi.scsi_target where target_id=0x650060"
{ "values": {
        "1": {
                "port": "fc3/17",
                "target_id": "0x650060",
                "total_write_io_count": "0",
                "total_write_io_bytes": "0",
                "total_time_metric_based_write_io_count": "0",
                "write_io_rate": "0",
```

```
            "peak_write_io_rate": "0",
            "write_io_bandwidth": "0",
            "peak_write_io_bandwidth": "0",
            "write_io_size_min": "0",
            "write_io_size_max": "0",
            "write_io_completion_time_min": "0",
            "write_io_completion_time_max": "0",
            "write_io_initiation_time_min": "0",
            "write_io_initiation_time_max": "0",
            "write_io_inter_gap_time_min": "0",
            "write_io_inter_gap_time_max": "0",
            "sampling_start_time": "1531204359",
            "sampling_end_time": "1531215464"
    }
```

This example shows the output after purging the flow metrics:

**Note** Only the *port* key value is allowed with the where clause for purging metrics.

```
switch# purge analytics query "select all from fc-scsi.scsi_target where port=fc3/17"
switch# show analytics query "select all from fc-scsi.scsi_target where port=fc3/17"
Table is empty for query "select all from fc-scsi.scsi_target where port=fc3/17"
```

# Using the ShowAnalytics Overlay CLI

Overlay CLI is used to interpret the analytics data that is in JSON format in a user-friendly tabular format. Overlay CLI has a "Linux like" syntax and uses the inbuilt NX-OS python interpreter to execute a script to convert the JSON output of the pull query into a tabular format. Currently, only a small subset of the metrics are displayed.

To display the analytics information in a tabular format, run this command:

switch# **ShowAnalytics** *–help*

```
switch# ShowAnalytics -help
usage: analytics [-h] [--version] [--info] [--errors] [--initiator-itl] [--target-itl]
[--interface INTERFACE] [--vsan VSAN] [--target TARGET] [--initiator INITIATOR] [--lun LUN]

analytics optional arguments:
  -h, --help            show this help message and exit
  --version             version
  --info                --info | --errors mandatory
  --errors              --info | --errors mandatory
  --initiator-itl       --initiator-itl | --target-itl mandatory
  --target-itl          --initiator-itl | --target-itl mandatory
  --interface INTERFACE
                        fc interface
  --vsan VSAN           vsan
  --target TARGET       target FCID
  --initiator INITIATOR
                        initiator FCID
  --lun LUN             lun
```

***Table 5: Syntax Description***

| | |
|---|---|
| *-h*, *--help* | Provides information about the list of available keywords and arguments. |
| *--version* | Displays the SAN Analytics version. |
| *--info* | Displays information specific to a view type. |
| *--errors* | Displays errors specific to a view type. |
| *--initiator-itl* | Displays the SAN Analytics information of the initiator-target-LUN flow. |
| *--target-itl* | Displays the SAN Analytics information of the target ITL flow. |
| *--interface port/slot* | Displays the SAN Analytics information for a specific interface. |
| *--vsan id* | Displays the SAN Analytics information for a specific VSAN. |
| *--target id* | Displays the SAN Analytics information for a specific target. |
| *--initiator id* | Displays the SAN Analytics information for a specific initiator. |
| *--lun id* | Displays the SAN Analytics information for a specific LUN. |

# Examples: Using the ShowAnalytics Overlay CLI

This example shows how to get information about using the overlay CLI:

```
switch# ShowAnalytics -help
usage: analytics [-h] [--version] [--info] [--errors] [--initiator-itl] [--target-itl]
[--interface INTERFACE] [--vsan VSAN] [--target TARGET] [--initiator INITIATOR]
[--lun LUN] analytics optional arguments:
  -h, --help            show this help message and exit
  --version             version
  --info                --info | --errors mandatory
  --errors              --info | --errors mandatory
  --initiator-itl       --initiator-itl | --target-itl mandatory
  --target-itl          --initiator-itl | --target-itl mandatory
  --interface INTERFACE
                        fc interface
  --vsan VSAN           vsan
  --target TARGET       target FCID
  --initiator INITIATOR
                        initiator FCID
  --lun LUN             lun
```

This example shows how to display the overlay CLI version:

```
switch# ShowAnalytics --version
analytics 1.0
```

This example displays the flow metrics of an initiator ITL:

```
switch# ShowAnalytics --info --initiator-itl
 Interface fc12/15
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                             | Avg IOPS |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write| Read|Write    | Read|Write    |
|                                        |          |               |               |
| 4|0x650060|0x650040|0000-0000-0000-0000 | 2355|2311|9646080|9467904| 282|248       |
| 4|0x650060|0x650040|0001-0000-0000-0000 | 2313|2330|9477120|9543680| 282|236       |
| 4|0x650060|0x650040|0002-0000-0000-0000 | 2324|2337|9519104|9574400| 298|236       |
| 4|0x650061|0x650041|0001-0000-0000-0000 | 2335|2294|9567232|9397248| 274|254       |
| 4|0x650061|0x650041|0000-0000-0000-0000 | 2283|2333|9351168|9559040| 284|246       |
| 4|0x650061|0x650041|0002-0000-0000-0000 | 2302|2354|9431040|9645056| 289|252       |
+----------------------------------------+----------+---------------+---------------+
Interface fc3/15
+----------------------------------------+----------+----------------+---------------+
| VSAN|I|T|L                             | Avg IOPS |Avg Thput (B/s) | Avg ECT (usec)|
+----------------------------------------+----------+----------------+---------------+
|                                        |Read|Write| Read|Write     | Read|Write    |
|                                        |          |                |               |
| 1|0x220380|0x2203e0|0001-0000-0000-0000 |2315|2358 | 9482240|9661440| 101|62        |
| 1|0x220380|0x2203e0|0000-0000-0000-0000 |2330|2355 | 9545728|9647104| 102|62        |
| 1|0x220380|0x2203e0|0002-0000-0000-0000 |2334|2335 | 9560064|9567232| 102|61        |
| 1|0x220381|0x2203e2|0001-0000-0000-0000 |2414|2379 | 9888768|9744384| 99|63         |
| 1|0x220381|0x2203e2|0002-0000-0000-0000 |2381|2323 | 9753600|9518080| 101|64        |
| 1|0x220381|0x2203e2|0000-0000-0000-0000 |2315|2354 | 9483264|9643008| 104|64        |
+----------------------------------------+----------+----------------+---------------+
```

This example displays the flow metrics of a target ITL:

```
switch# ShowAnalytics --info --target-itl
Interface fc12/16
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                              |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 1|0x220381|0x2203e2|0000-0000-0000-0000 |2334|2308 |9562112|9456640| 135|94        |
| 1|0x220381|0x2203e2|0002-0000-0000-0000 |2339|2375 |9581568|9731072| 137|92        |
| 1|0x220380|0x2203e0|0000-0000-0000-0000 |2353|2299 |9637888|9418752| 135|97        |
| 1|0x220380|0x2203e0|0002-0000-0000-0000 |2303|2363 |9434112|9678848| 136|92        |
| 1|0x220380|0x2203e0|0001-0000-0000-0000 |2315|2335 |9485312|9565184| 137|95        |
| 1|0x220381|0x2203e2|0001-0000-0000-0000 |2311|2375 |9465856|9731072| 135|96        |
+----------------------------------------+----------+---------------+---------------+

 Interface fc3/16
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                              |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 4|0x650060|0x650040|0002-0000-0000-0000 |2299|2367 |9417728|9696256| 247|199       |
| 4|0x650060|0x650040|0000-0000-0000-0000 |2353|2310 |9637888|9462784| 250|204       |
| 4|0x650060|0x650040|0001-0000-0000-0000 |2320|2326 |9503744|9529344| 247|209       |
| 4|0x650061|0x650041|0002-0000-0000-0000 |2327|2320 |9532416|9504768| 236|205       |
| 4|0x650061|0x650041|0000-0000-0000-0000 |2322|2318 |9513984|9494528| 244|209       |
| 4|0x650061|0x650041|0001-0000-0000-0000 |2311|2333 |9467904|9559040| 257|198       |
+----------------------------------------+----------+-----------------+-------------+
```

This example displays the flow metrics of VSAN 1 of an initiator ITL:

```
switch# ShowAnalytics --info --initiator-itl --vsan 1
 Interface fc3/15
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                              |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 1|0x220380|0x2203e0|0001-0000-0000-0000 |2317|2324 |9491456|9521152| 106|68        |
| 1|0x220380|0x2203e0|0000-0000-0000-0000 |2337|2300 |9573376|9420800| 107|68        |
| 1|0x220380|0x2203e0|0002-0000-0000-0000 |2331|2317 |9547776|9491456| 107|66        |
| 1|0x220381|0x2203e2|0001-0000-0000-0000 |2316|2331 |9487360|9547776| 104|69        |
| 1|0x220381|0x2203e2|0002-0000-0000-0000 |2316|2332 |9486336|9553920| 106|69        |
| 1|0x220381|0x2203e2|0000-0000-0000-0000 |2313|2298 |9474048|9412608| 109|69        |
+----------------------------------------+----------+---------------+---------------+
```

This example displays the flow metrics of interface fc12/19 of a target ITL:

```
switch# ShowAnalytics --info --target-itl --interface fc12/19
 Interface fc12/19
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                              |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 4|0x650020|0x650000|0000-0000-0000-0000 |0|7486    |0|15331840     | 0|211         |
+----------------------------------------+----------+---------------+---------------+
```

This example displays the flow metrics of target ID 0x2203e2 of a target ITL:

```
switch# ShowAnalytics --info --target-itl --target 0x2203e2
 Interface fc12/16
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                             |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 1|0x220381|0x2203e2|0000-0000-0000-0000 |2323|2301 |9518080|9427968| 145|105       |
| 1|0x220381|0x2203e2|0002-0000-0000-0000 |2293|2343 |9395200|9597952| 148|102       |
| 1|0x220381|0x2203e2|0001-0000-0000-0000 |2282|2341 |9349120|9591808| 145|106       |
+----------------------------------------+----------+---------------+---------------+
```

This example displays the flow metrics of initiator ID 0x220381 and LUN ID 0002-0000-0000-0000 of a target ITL:

```
switch# ShowAnalytics --info --target-itl --initiator 0x220381 --lun 0002-0000-0000-0000
Interface fc12/16
+----------------------------------------+----------+---------------+---------------+
| VSAN|I|T|L                             |Avg IOPS  |Avg Thput (B/s)| Avg ECT (usec)|
+----------------------------------------+----------+---------------+---------------+
|                                        |Read|Write|Read|Write     | Read|Write    |
|                                        |          |               |               |
| 1|0x220381|0x2203e2|0002-0000-0000-0000 |2319|2340 |9500672|9585664| 150|104       |
+----------------------------------------+----------+---------------+---------------+
```

This example displays the errors on interface fc12/21 of an initiator ITL:

```
switch# ShowAnalytics --errors --initiator-itl --interface fc12/21
Interface fc12/21
+----------------------------------------+---------------------+-----------------+
| VSAN|I|T|L                             | Total SCSI Failures | Total FC Aborts |
+----------------------------------------+---------------------+-----------------+
|                                        | Read|Write          | Read|Write      |
|                                        |                     |                 |
| 4|0x650020|0x650000|0000-0000-0000-0000 | 0|0                 | 0|18            |
+----------------------------------------+---------------------+-----------------+
```

# Displaying Congestion Drops Per Flow

The SAN Analytics feature displays packet timeout drops on a per-flow basis. The number of packets dropped along with the time stamp for ports is displayed.

To display the packet drops on a per-flow basis, run this command:

switch# **show analytics type fc-scsi flow congestion-drops**

## Examples: Displaying Congestion Drops Per Flow

This example shows the interfaces where timeout drops occurred due to congestion in a network:

```
switch# show analytics type fc-scsi flow congestion-drops

|==========================================================================================|
| Source       |Destination            | Congestion    |     Timestamp                 |
| INTF  | VSAN |  FCID     |  FCID     | Drops(delta)  |                               |
|==========================================================================================|
| fc2/13| 0002 | 0x9900E1  | 0x640000  |  00000105     | 1. 09/13/17 11:09:48.762      |
| fc2/13| 0002 | 0x9900E1  | 0x640000  |  00000002     | 2. 09/13/17 09:05:39.527      |
| fc2/13| 0002 | 0x990000  | 0x640020  |  00000002     | 3. 09/13/17 09:05:39.527      |
|==========================================================================================|
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000084     | 1. 09/12/17 08:17:11.905      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000076     | 2. 09/12/17 05:50:37.721      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000067     | 3. 09/12/17 03:24:03.319      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000088     | 4. 09/12/17 00:57:28.019      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000088     | 5. 09/11/17 22:30:53.723      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000086     | 6. 09/11/17 20:04:18.001      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000026     | 7. 09/11/17 17:37:24.273      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000076     | 8. 09/11/17 15:10:50.240      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000074     | 9. 09/11/17 12:44:15.866      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000087     |10. 09/11/17 10:17:41.402      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000086     |11. 09/11/17 07:51:10.412      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000084     |12. 09/11/17 05:24:35.981      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000083     |13. 09/11/17 02:58:01.067      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000086     |14. 09/11/17 00:31:26.709      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000079     |15. 09/10/17 22:04:51.399      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000084     |16. 09/10/17 19:38:17.217      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000082     |17. 09/10/17 17:11:42.594      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000086     |18. 09/10/17 14:44:52.786      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000089     |19. 09/10/17 12:18:18.394      |
| fc2/31| 0002 | 0x640000  | 0x9900E1  |  00000087     |20. 09/10/17 09:51:44.067      |
|==========================================================================================|
```

# Verifying SAN Analytics

This example shows the list of interfaces that have the SAN Analytics feature enabled:

```
switch# show running-config analytics
!Command: show running-config analytics
!Running configuration last done at: Fri Jun 29 09:03:09 2018
!Time: Fri Jun 29 13:00:41 2018

version 8.3(1)
feature analytics

analytics query "Select all from fc-scsi.port" name port type periodic interval 30
analytics query "Select all from fc-scsi.port where port=fc1/1" name portfc1_1 type periodic
 interval 30  differential
analytics query "Select all from fc-scsi.port where port=fc1/32" name portfc1_32 type
periodic interval 30  differential
analytics query "select port, vsan, app_id, initiator_id, target_id, lun,
active_io_read_count, active_io_write_count, total_read_io_count, total_write_io_count,
total_time_metric_
based_read_io_count, total_time_metric_based_write_io_count,total_read_io_time,
total_write_io_time, total_read_io_initiation_time,
total_write_io_initiation_time,total_read_io_byt
es, total_write_io_bytes, total_time_metric_based_read_io_bytes,
total_time_metric_based_write_io_bytes, read_io_rate, write_io_rate, read_io_bandwidth,
write_io_bandwidth,read_io_
size_min, read_io_size_max, write_io_size_min, write_io_size_max,read_io_completion_time_min,
 read_io_completion_time_max, write_io_completion_time_min, write_io_completion_time_ma
x,read_io_initiation_time_max, write_io_initiation_time_max, read_io_aborts,
write_io_aborts,read_io_failures, write_io_failures, read_io_timeouts, write_io_timeouts
from fc-scsi.s
csi_initiator_itl_flow" name dcnminitITL type periodic interval 30

interface fc1/2
  analytics type fc-scsi

interface fc1/4
  analytics type fc-scsi

interface fc1/6
  analytics type fc-scsi

interface fc1/7
  analytics type fc-scsi

interface fc1/8
  analytics type fc-scsi

interface fc1/9
  analytics type fc-scsi

interface fc1/10
  analytics type fc-scsi

interface fc1/11
  analytics type fc-scsi
```

This example shows the list of configured push queries that are installed on a switch:

```
switch(config)# show analytics query all
Total queries:7
============================
Query Name      :init
Query String    :select all from fc-scsi.scsi_initiator
Query Type      :periodic, interval 30

Query Name      :targettl
Query String    :select all from fc-scsi.scsi_target_tl_flow
Query Type      :periodic, interval 30
Query Options   :differential clear

Query Name      :port
Query String    :select all from fc-scsi.logical_port
Query Type      :periodic, interval 30

Query Name      :targetit
Query String    :select all from fc-scsi.scsi_target_it_flow
Query Type      :periodic, interval 30

Query Name      :targetitl
Query String    :select all from fc-scsi.scsi_target_itl_flow
Query Type      :periodic, interval 30
Query Options   :differential clear

Query Name      :inititl
Query String    :select all from fc-scsi.scsi_initiator_itl_flow
Query Type      :periodic, interval 30

Query Name      :initit
Query String    :select all from fc-scsi.scsi_initiator_it_flow
Query Type      :periodic, interval 30
```

This example shows how to check the port sampling status and the instantaneous NPU load:

**Note** The star symbol (*) next to a port indicates that the port is currently being sampled.

```
switch# show analytics port-sampling module 1
Sampling Window Size: 1
Rotation Interval: 30
NPU Load: 0%
==========================================================================
  Port                Monitored Start Time           Monitored End Time
==========================================================================
  fc1/4               07/09/18 - 08:33:20            07/09/18 - 08:33:50
  fc1/6               07/09/18 - 08:33:50            07/09/18 - 08:34:20
  fc1/7               07/09/18 - 08:34:20            07/09/18 - 08:34:50
  fc1/8               07/09/18 - 08:34:50            07/09/18 - 08:35:20
  fc1/10              07/09/18 - 08:35:20            07/09/18 - 08:35:50
  fc1/11              07/09/18 - 08:35:50            07/09/18 - 08:36:20
  fc1/12              07/09/18 - 08:36:20            07/09/18 - 08:36:50
  fc1/14              07/09/18 - 08:36:50            07/09/18 - 08:37:20
  fc1/15              07/09/18 - 08:37:20            07/09/18 - 08:37:50
  fc1/16              07/09/18 - 08:37:50            07/09/18 - 08:38:20
  fc1/18*             07/09/18 - 08:38:20            -
  fc1/20              07/09/18 - 08:28:50            07/09/18 - 08:29:20
  fc1/22              07/09/18 - 08:29:20            07/09/18 - 08:29:50
  fc1/23              07/09/18 - 08:29:50            07/09/18 - 08:30:20
```

```
  fc1/24             07/09/18 - 08:30:20            07/09/18 - 08:30:50
  fc1/26             07/09/18 - 08:30:50            07/09/18 - 08:31:20
  fc1/27             07/09/18 - 08:31:20            07/09/18 - 08:31:50
  fc1/28             07/09/18 - 08:31:50            07/09/18 - 08:32:20
  fc1/30             07/09/18 - 08:32:20            07/09/18 - 08:32:50
  fc1/32             07/09/18 - 08:32:50            07/09/18 - 08:33:20
```

This example shows the output of a push query that has already been configured:

```
switch# show analytics query name iniitl result
{ "values": {
        "1": {
                "port": "fc1/6",
                "vsan": "10",
                "app_id": "255",
                "initiator_id": "0xe800a0",
                "target_id": "0xd601e0",
                "lun": "0000-0000-0000-0000",
                "active_io_read_count": "0",
                "active_io_write_count": "7",
                "total_read_io_count": "0",
                "total_write_io_count": "1008608573",
                "total_seq_read_io_count": "0",
                "total_seq_write_io_count": "1",
                "total_read_io_time": "0",
                "total_write_io_time": "370765952314",
                "total_read_io_initiation_time": "0",
                "total_write_io_initiation_time": "52084968152",
                "total_read_io_bytes": "0",
                "total_write_io_bytes": "2065630357504",
                "total_read_io_inter_gap_time": "0",
                "total_write_io_inter_gap_time": "16171468343166",
                "total_time_metric_based_read_io_count": "0",
                "total_time_metric_based_write_io_count": "1008608566",
                "total_time_metric_based_read_io_bytes": "0",
                "total_time_metric_based_write_io_bytes": "2065630343168",
                "read_io_rate": "0",
                "peak_read_io_rate": "0",
                "write_io_rate": "16070",
                "peak_write_io_rate": "32468",
                "read_io_bandwidth": "0",
                "peak_read_io_bandwidth": "0",
                "write_io_bandwidth": "32912384",
                "peak_write_io_bandwidth": "66494976",
                "read_io_size_min": "0",
                "read_io_size_max": "0",
                "write_io_size_min": "2048",
                "write_io_size_max": "2048",
                "read_io_completion_time_min": "0",
                "read_io_completion_time_max": "0",
                "write_io_completion_time_min": "111",
                "write_io_completion_time_max": "9166",
                "read_io_initiation_time_min": "0",
                "read_io_initiation_time_max": "0",
                "write_io_initiation_time_min": "36",
                "write_io_initiation_time_max": "3265",
                "read_io_inter_gap_time_min": "0",
                "read_io_inter_gap_time_max": "0",
                "write_io_inter_gap_time_min": "100",
                "write_io_inter_gap_time_max": "1094718",
                "peak_active_io_read_count": "0",
                "peak_active_io_write_count": "23",
                "read_io_aborts": "0",
```

```
            "write_io_aborts": "0",
            "read_io_failures": "0",
            "write_io_failures": "0",
            "read_io_timeouts": "0",
            "write_io_timeouts": "0",
            "read_io_scsi_check_condition_count": "0",
            "write_io_scsi_check_condition_count": "0",
            "read_io_scsi_busy_count": "0",
            "write_io_scsi_busy_count": "0",
            "read_io_scsi_reservation_conflict_count": "0",
            "write_io_scsi_reservation_conflict_count": "0",
            "read_io_scsi_queue_full_count": "0",
            "write_io_scsi_queue_full_count": "0",
            "sampling_start_time": "1529993232",
            "sampling_end_time": "1529993260"
    },
    "2": {
            "port": "fc1/6",
            "vsan": "10",
            "app_id": "255",
            "initiator_id": "0xe800a1",
            "target_id": "0xd601e1",
            "lun": "0000-0000-0000-0000",
            "active_io_read_count": "0",
            "active_io_write_count": "8",
            "total_read_io_count": "0",
            "total_write_io_count": "1004271260",
            "total_seq_read_io_count": "0",
            "total_seq_write_io_count": "1",
            "total_read_io_time": "0",
            "total_write_io_time": "370004164726",
            "total_read_io_initiation_time": "0",
            "total_write_io_initiation_time": "51858511487",
            "total_read_io_bytes": "0",
            "total_write_io_bytes": "2056747540480",
            "total_read_io_inter_gap_time": "0",
            "total_write_io_inter_gap_time": "16136686881766",
            "total_time_metric_based_read_io_count": "0",
            "total_time_metric_based_write_io_count": "1004271252",
            "total_time_metric_based_read_io_bytes": "0",
            "total_time_metric_based_write_io_bytes": "2056747524096",
            "read_io_rate": "0",
            "peak_read_io_rate": "0",
            "write_io_rate": "16065",
            "peak_write_io_rate": "16194",
            "read_io_bandwidth": "0",
            "peak_read_io_bandwidth": "0",
            "write_io_bandwidth": "32901632",
            "peak_write_io_bandwidth": "33165824",
            "read_io_size_min": "0",
            "read_io_size_max": "0",
            "write_io_size_min": "2048",
            "write_io_size_max": "2048",
            "read_io_completion_time_min": "0",
            "read_io_completion_time_max": "0",
            "write_io_completion_time_min": "114",
            "write_io_completion_time_max": "9019",
            "read_io_initiation_time_min": "0",
            "read_io_initiation_time_max": "0",
            "write_io_initiation_time_min": "37",
            "write_io_initiation_time_max": "3158",
            "read_io_inter_gap_time_min": "0",
            "read_io_inter_gap_time_max": "0",
            "write_io_inter_gap_time_min": "101",
```

```
                    "write_io_inter_gap_time_max": "869035",
                    "peak_active_io_read_count": "0",
                    "peak_active_io_write_count": "19",
                    "read_io_aborts": "0",
                    "write_io_aborts": "0",
                    "read_io_failures": "0",
                    "write_io_failures": "0",
                    "read_io_timeouts": "0",
                    "write_io_timeouts": "0",
                    "read_io_scsi_check_condition_count": "0",
                    "write_io_scsi_check_condition_count": "0",
                    "read_io_scsi_busy_count": "0",
                    "write_io_scsi_busy_count": "0",
                    "read_io_scsi_reservation_conflict_count": "0",
                    "write_io_scsi_reservation_conflict_count": "0",
                    "read_io_scsi_queue_full_count": "0",
                    "write_io_scsi_queue_full_count": "0",
                    "sampling_start_time": "1529993232",
                    "sampling_end_time": "1529993260"
          }
}}
```


**Note** The output of these queries are in JSON format.