



NVIDIA UFM Telemetry Documentation v1.23.1

Table of contents

Release Notes	4
New Features in v1.23.1	4
System Requirements	4
Bug Fixes in This Release	4
Known Issues in This Release	5
Overview	6
Software Management	7
Data Collection	22
Fluent Bit Export	35
Settings and Configuration	55
Port Counters	59
Events	64
Prometheus Endpoint Support	68
Distributed Telemetry - Switch Telemetry Agent	80
Supported Telemetry Fields	86
Ports Counters Info	86
Cable Info	90
Switch Info	98
PCI Info	98
NVLink Info	102
Documentation History	103
Document Revision History	103

Release Notes History	104
Bug Fixes History	104

About This Document

NVIDIA® UFM® Telemetry platform provides network validation tools to monitor network performance and conditions, capturing and streaming rich real-time network telemetry information along with application workload usage to an on-premise or cloud-based database for further analysis.

Technical Support

Customers who purchased NVIDIA products directly from NVIDIA are invited to contact us through the following methods:

- E-mail: enterprisesupport@nvidia.com
- Enterprise Support page: [Enterprise Support Services](#)

Customers who purchased NVIDIA M-1 Global Support Services, please see your contract for details regarding Technical Support.

Customers who purchased NVIDIA products through an NVIDIA-approved reseller should first seek assistance through their reseller.

Document Revision History

For the list of changes made to this document, refer to [Document Revision History](#).

Release Notes

These release notes pages provide information for NVIDIA UFM Telemetry such as changes and new features and bug fixes.

New Features in v1.23.1

- Added switch port 0 in reports alongside physical ports.
- NVIDIA Quantum-4: Added support to all available link speeds.
- NVIDIA Quantum-3 : Added AMBER support for CPO systems.
- Introduced host counters for retransmission errors and timeouts.
- Added support for NVIDIA NVLink bi-directional 2x mode in Fabric discovery.

System Requirements

Platform	Type and Version
OS and Kernel	<ul style="list-style-type: none">• RedHat 8.X• RedHat 9.X• Ubuntu18• Ubuntu20• Ubuntu22• Ubuntu24
CPU	x86_64
OFED	MLNX_OFED 5.X

Bug Fixes in This Release

Ref. #	Description
4574975	Description: Corrupted lookup for <code>phy_mgr_fsm_state</code> may cause corrupted csv
	Keywords: Corrupted csv, <code>phy_mgr_fsm_state</code>
	Discovered in Release: v1.22.1
4571568	Description: Fixed issue with negative congestion BW in XDR
	Keywords: Negative values
	Discovered in Release: v1.22.1
4184583	Description: Fixed issues with missing Labels metadata on xcset
	Keywords: Labels, xcset
	Discovered in Release: v1.22.1
4661792	Description: Resolved an issue where the timestamp continued updating even after the port became unreachable.
	Keywords: Timestamp, Unreachable Port
	Discovered in Release: v1.22.1
4639292	Description: Resolved an issue where XDR was missing aggregation ports from the secondary telemetry endpoint.
	keywords: Missing Aggregation Ports, XDR
	Discovered in Release: v1.22.1
4664041	Description: Resolved an issue where active ports were incorrectly displayed as down due to <code>port_state_update</code> .
	keywords: Port state, Down Port, Wrong State
	Discovered in Release: v1.22.1
4693207	Description: Fixed issue where telemetry could crash after a switch reboot
	keywords: Crash, Switch Reboot
	Discovered in Release: v1.22.1

Known Issues in This Release

N/A

Overview

NVIDIA® UFM® Telemetry platform provides network validation tools to monitor network performance and conditions, and to capture and stream rich real-time network telemetry information and application workload usage to an on-premise or cloud-based database for further analysis.

UFM Telemetry can be used to monitor the basic fabric port counters and network statistics at a relatively high rate, or a more exhaustive set of performance metrics at a lower rate (referred to as Bringup mode). It can be configured to save collected data to disk, to stream via a Fluent forward protocol, or to make the data available via an http endpoint in csv or Prometheus format.

UFM Telemetry is packaged both as a docker image and as a bare metal tarball package.

Software Management

Deploying UFM Telemetry

Deploying UFM Telemetry can be done in the following modes:

- [Bare Metal - Bringup Mode](#)
- [Docker Container Mode](#)
- [Docker Container Mode - High Availability](#)
- [Bare Metal Mode](#)
- [Bare Metal Mode - High Availability](#)

Bare Metal - Bringup Mode

NVIDIA UFM Telemetry can be obtained as a tarball for installation on a Linux machine with all prerequisites installed.

To deploy the UFM Telemetry in Bringup mode, perform the following steps:

1. Make sure the following prerequisites are installed:
 1. Python3
 2. Python3-venv
 3. Supervisor
2. Copy the tarball package to the targeted location.
3. Extract the package.

```
tar -xf ufm_telemetry-<version>.tar.gz
```

4. Start collection.

```
./bin/run_bringup .sh
```

```
CollectX: collection_start
```

This collects port counter and cable data every minute, uses HCA mlx5_0 and writes data to `./collection_data/clx-bringup-X` for a period of 24hrs.

```
CollectX: help collection_start
```

```
Usage:
```

defaults	Description	options	
-----			----
	collection_start	time duration=n [s m h d]	24h
Session duration			
		sample_rate=n [s m h d]	60
seconds	Data sample rate		
		guids=[guid_list guid_file]	None
Target devices	guid		
		counter_set=[file.xcset]	None
Counter list to be collected			
		hca=hca_name	
mlx5_0	Device to access the fabric		
		cable cable_info=[yes no once]	yes
Collect cable info			
		nvlink_info=[yes no]	no
Collect NVLink info			
		disconnected_cable=[yes no]	no
Collect disconnected cables info			

	reset_counters=t	false
Reset counters of fabric devices		
	compress_data=[yes no]	yes
Compress files (if write_files=true)		
	mads_retries=n	2
Set number of retries for MADs		
	mads_timeout=n (msec)	500
Set timeout for MADs		
	force_hca=t	f
Avoid HCA state check		

Docker Container Mode

NVIDIA UFM Telemetry is packaged as a docker image that should be loaded and deployed on a Linux machine with docker installed. This section describes how to deploy the UFM Telemetry docker image on a Linux machine.

To deploy the UFM telemetry, perform the following steps:

1. Make sure that docker is installed on the Linux machine.

```
[root@r-ufm ~]# docker -version
```

2. Start the docker service.

```
[root@r-ufm ~]# sudo service docker start
```

3. Pull the image.

```
[root@r-ufm ~]# export image=mellanox/ufm-telemetry:<version>
```

```
[root@r-ufm ~]# sudo docker pull $image
```

4. Create the default .ini files and place them in the local directory mapped to /config in the container and initialize the container configuration.

```
root@r-ufm ~]# sudo docker run -v /opt/ufm-  
telemetry/conf:/config --rm -d $image  
/get_collectx_configs.sh  
"sample_rate=300;hca=mlx5_0;cable_info_schedule=1/00:00,3/00:00"
```

Note

This collects port counter data every 5 minutes and uses HCA mlx5_0. It also collects cable info on the 1st, 3rd, and 5th day of the week at midnight, where:

- sample_rate: Frequency of collecting port counters
- hca: Card to use
- cable_info_schedule: Time of collecting cable info data (optional)

5. Create a container of UFM telemetry.

```
root@r-ufm ~]# sudo docker run --net=host --uts=host --  
ipc=host \  
    --ulimit stack=67108864 --ulimit memlock=-1 \  
    --security-opt seccomp=unconfined --cap-  
add=SYS_ADMIN \  
    --device=/dev/infiniband/ -v "/opt/ufm-  
telemetry/conf:/config" -v "/tmp/data:/data" -v
```

```
"/opt/ufm/files/licenses:/opt/ufm/files/licenses/" --rm --  
name ufm-telemetry -d $image
```

6. Verify that UFM Telemetry is running.

1. Make sure the UFM Telemetry container is up.

```
[root@r-ufm ~]# docker ps
```

2. If the container name exists, access the shell of the container.

```
[root@r-ufm ~]# docker exec -it ufm-telemetry bash
```

3. Review your configurations under

```
/config/launch_ibdiagnet_config.ini.
```

7. View the UFM Telemetry configuration files.

```
root@ r-ufm ~]# ls -l /config/  
-rw-r--r-- 1 3478 101 396 Apr 15 21:04 clx_config.ini  
-rw-r--r-- 1 3478 101 2987 Apr 15 21:04 collectx.ini  
-rw-r--r-- 1 3478 101 4257 Apr 15 21:04  
launch_ibdiagnet_config.ini  
-rw-r--r-- 1 3478 101 1912 Apr 16 12:03 supervisord.conf
```

8. To watch and review the execution of the various components, you can check the log files under `/var/log`. Each component has a dedicated log file. Running the "ls -l" command will display all files under the folder. The following output shows only the relevant log files (other files have been omitted).

```
[root@r-ufm ~]# ls -l /var/log
```

```
-rw-r--r-- 1 root root 128393 Apr  3 10:49
launch_cableinfo.log
-rw-r--r-- 1 root root    467 Apr  3 09:35
launch_compression.log
-rw-r--r-- 1 root root 194566 Apr  3 10:49
launch_ibdiagnet.log
-rw-r--r-- 1 root root    798 Apr  3 09:35
launch_retention.log
-rw-r--r-- 1 root root    1729 Apr  3 09:56 supervisord.log
```

9. To exit the UFM Telemetry docker context, run "exit" to return to the Linux machine context.

10. To access the UFM Telemetry CLI, run the following command on the Linux machine:

```
[root@r-ufm ~]# docker exec -it ufm-telemetry clxcli
```

11. For settings and configuration instructions, see [Settings and Configuration](#).

Docker Container Mode - High Availability

Requirements:

- An important requirement for the HA solution is to prepare a dedicated partition for DRBD to work with. Example of such a requirement: /dev/sda4.
- Install pcs and drbd-utils on both servers (using “yum” or “apt-get install”, based on your OS).

Note

On RH/CentOS, please run “yum install pcs drbd84-utils kmod-drbd84.

Procedure:

1. Load (pull) the latest UFM Telemetry Docker image on both servers.

```
docker pull mellanox/ufm-telemetry:latest
```

2. Run the Telemetry configuration command on both servers.

```
docker run --rm -i --name=config-telemetry \  
-v /opt/ufm-telemetry/conf:/config \  
-v /etc/systemd/system:/etc/systemd/system \  
-v /var/run/docker.sock:/var/run/docker.sock \  
mellanox/ufm-telemetry:latest \  
/get_collectx_configs.sh \  
--gen_service \  
--config=ufm_telemetry
```

3. Refresh systemd on both servers:

```
systemctl daemon-reload
```

4. Create the `/opt/ufm-telemetry/licenses/` directory on the master server and copy the UFM Telemetry license file there.
5. Download UFM-HA Package on both servers from [this link](#).
6. Extract the HA package to `/tmp/`, and from there, run the installation command on both servers as follows:

Note

In the below commands, "disk", the partition name, is assumed as /dev/sda4.

```
./install -l /opt/ufm-telemetry/ -d /dev/sda4 -p telemetry
```

7. Run the UFM-HA configuration command **ONLY** on the master server, as follows:

```
configure_ha_nodes.sh \  
--cluster-password 12345678 \  
--master-ip 192.168.10.1 \  
--standby-ip 192.168.10.2 \  
--virtual-ip 192.168.10.5
```

Note

The `cluster-password` must be at least 8 characters long.

Note

Change the values of in the above command with your server' information.

8. Start UFM Telemetry HA cluster. Run:

```
ufm_ha_cluster start
```

Bare Metal Mode

NVIDIA® UFM® Telemetry can be obtained as a tarball for installation on a Linux machine with all prerequisites installed.

To deploy the UFM Telemetry:

1. Ensure the following prerequisites are installed:

1. Python3
2. Python3-venv
3. Supervisor

2. Copy the tarball package to the target location.

3. Extract package.

```
tar -xf ufm_telemetry-<version>.tar.gz
```

4. Initialize and configure.

```
./bin/initialize_telemetry.sh --telemetry-dir  
/tmp/ufm_telemetry --config  
"hca=mlx5_0;sample_rate=300;data_dir=/tmp/clx_data;plugin_env_
```



This collects port counter data every 5 minutes, and uses HCA mlx5_0 and writes data to /tmp/clx_data.

5. Start data collection.

```
supervisord --config /tmp/ufm_telemetry/conf/supervisord.conf
```

Bare Metal Mode - High Availability

NVIDIA® UFM® Telemetry can be obtained as a tarball for installation on a Linux machine with all prerequisites installed.

To deploy the UFM Telemetry:

1. Ensure the following prerequisites are installed:
 1. Python3
 2. Python3-venv
 3. Supervisor
2. Copy the tarball package to the target location.
3. Extract package.

```
tar -xf ufm_telemetry -<version>.tar.gz
```

4. Initialize and configure.

```
./bin/initialize_telemetry.sh --telemetry-dir  
/tmp/ufm_telemetry --config
```

```
"hca=mlx5_0;sample_rate=300;data_dir=/tmp/clx_data;plugin_env_0
--gen_systemd_service
```

Note

This collects port counter data every 5 minutes, and uses HCA mlx5_0 and writes data to /tmp/clx_data.

5. Download UFM-HA Package on both servers from [this link](#).
6. Extract the HA package to `/tmp/`, and from there, run the installation command on both servers as follows:

Note

In the below commands, "disk", the partition name, is assumed as /dev/sda4.

```
./install -l /opt/ufm-telemetry/ -d /dev/sda4 -p telemetry
```

7. Run the UFM-HA configuration command **ONLY** on the master server, as follows:

```
configure_ha_nodes.sh \  
--cluster-password 12345678 \  
--master-ip 192.168.10.1 \  
--standby-ip 192.168.10.2 \  
--virtual-ip 192.168.10.5
```

Note

The `cluster-password` must be at least 8 characters long.

Note

Change the values of in the above command with your server' information.

8. Start UFM Telemetry HA cluster. Run:

```
ufm_ha_cluster start
```

To check the status of your UFM Telemetry HA cluster, run:

```
ufm_ha_cluster status
```

To perform failover, run:

```
ufm_ha_cluster failover
```

To perform takeover, run:

```
ufm_ha_cluster takeover
```

Upgrading UFM Telemetry Software

Upgrading UFM Telemetry requires removing the previous package, pulling the new version of the UFM telemetry package, configuring the telemetry, and starting it from the new installation package.

The upgrade procedure can be done in the three modes:

- [Bare Metal - Bringup Mode](#)
- [Docker Container Mode](#)
- [Bare Metal Mode](#)

Bare Metal - Bringup Mode

1. Stop previous collection. Run:

```
./bin/run_bringup.sh  
CollectX: collection_stop
```

2. Follow instructions described in [Deploying UFM Telemetry - Bare Metal Mode](#) with the new UFM Telemetry version.
3. If needed, apply the previous configuration changes.

Docker Container Mode

1. Stop the previous ufm-telemetry container.

```
[root@r-ufm ~]# docker stop ufm-telemetry
```

2. Pull the new UFM Telemetry image.

```
[root@r-ufm ~]# export image=mellanox/ufm-  
telemetry:rhel7.3_x86_64_ofed5.1-2.3.7_release_1.6_latest  
[root@r-ufm ~]# docker pull $image
```

3. Create a container for new UFM Telemetry.

```
[root@r-ufm ~]# docker run --net=host --uts=host --ipc=host \  
    --ulimit stack=67108864 --ulimit memlock=-1 \  
    --security-opt seccomp=unconfined --cap-  
add=SYS_ADMIN \  
    --device=/dev/infiniband/ -v "/opt/ufm-  
telemetry/conf:/config" -v "/tmp/data:/data" --rm --name ufm-  
telemetry -d $image
```

4. Configure the UFM Telemetry based on the new configurations.

```
[root@r-ufm ~]# docker run -v /opt/ufm-telemetry/conf:/config  
--rm -d $image /get_collectx_configs.sh  
sample_rate=300;hca=mlx5_0;cable_info_schedule=1/00:00,3/00:00,5/00:00
```

Bare Metal Mode

1. Stop previous collection. Run:

```
kill $SUPERVISORD_PID # send sigterm to the supervisord proc
```

2. Follow instructions described in [Deploying UFM Telemetry - Bringup Mode](#) with the new UFM Telemetry version.

3. If needed, apply the previous configuration changes.

Data Collection

NVIDIA® UFM® Telemetry uses the configuration file `launch_ibdiagnet_config.ini` to control the process of collecting the data. It collects two types of data: Cable info and port counters.

Port counters are collected periodically by setting the parameter `sample_rate` in seconds.

Bare Metal – Bringup Mode

The Bare Metal Bringup mode is the most common output format designed for debugging a cluster. The following command shows the help menu of the generated basic report command.

Description:

```
Dump basic results IB report for a given date or range of dates
```

Usage:

```
generate_basic_results_csv TIME [report_type=] [out=]  
[show_raw_data=]
```

TIME can be specified as:

date=

past=n[hours|days] : relative to the current time

on the server.

from= to=

[out=] to specify output file

```
[show_raw_data=t|f] boolean to show raw data as is. Default: f
```

Example:

```
generate_basic_results_csv past=10m out=basic_ib.csv
```

Bare Metal Mode

By default cable info data will not be collected. To enable its collection, add the following flag:

```
plugin_env_CLX_EXPORT_API_DISABLE_CABLEINFO=0
```

When enabled, cable info data is collected, by default, on every run. It is possible to change the collection frequency to be once every `num_iterations` using the following setting:

```
plugin_env_CLX_EXPORT_API_CABLE_RUN_ONCE=1
```

To work with the collected data, you may use the Telemetry CLI, which can be accessed as follows:

```
./bin/clxcli  
CollectX: set_data_root /tmp/clx_data  
CollectX: set_data_template {{year}}/{{month}}  
{{day}}/{{hash1023}}/{{source}}/{{tag}}/{{id}}.bin
```

Container Mode

Cable info data is collected based on a weekly schedule, set with the parameter `cable_info_schedule` . Time parameter is in the format "day/hrs:mins". For daily collection, it is "hrs:mins".

It is possible to collect the data multiple times during the week. To do that use a comma to separate the times at which collection is to take place. For example,

- `cable_info_schedule= 5/00:00` – collects cable info data on 5th day of the week at midnight
- `cable_info_schedule= 12:00` – collects cable info data midnight at 12:00 every day
- `cable_info_schedule= 5/00:00,12:00` – combines the previous two examples

To work with the collected data, you may use the Telemetry CLI, which can be accessed as follows:

```
[root@r-ufm145 ~]# docker exec -it ufm-telemetry clxcli
Read configuration from: /opt/mellanox/collectx/etc/collectx.ini
agx_data_root = /data
Loaded 2 schemas from /data/schema/schema*.json

CollectX:
```

Cable Info Data

The main commands to query and retrieve cable info data are `cable_times` and `cable_info` .

- `cables_times` – dump times and file names of cable info data files, and you can redirect the output to a file
- `cable_info` – dump cable info for a given date or range of dates

The following presents the help menu of the `cable_time` command:

```
CollectX: help cable_times
```

```
Usage:
```

```
    cable_times [TIME] [out=]
```

```
    [TIME] is one the following:
```

```
        date=
```

```
        past=n[hours|days]
```

```
Description:
```

```
    Dump times and file names of cable info data files
```

```
Examples:
```

```
    cable_times
```

```
    cable_times date=jun04
```

```
    cable_times past=15d out=out.csv
```

Example for `cable_time` command:

```
CollectX: cable_times
```

```
Opened 202 files in 0.05 seconds
```

```
Cable
```

```
-----
```

idx	Date Time	Filename
---	-----	-----
1	2020-07-26 04:13	/.../cables_1595725983912963.bin
3	2020-07-26 04:28	/.../cables_1595726884030804.bin

Help menu of `cable_info` command:

CollectX: help cable_info

Usage:

```
cable_info [TIME] [out=]
```

[TIME] is one of the following:

```
last  
date=
```

past=n[hours|days]

[out=] is to specify output file (optional)

Description:

Dump cable info for a given date or range of dates.

If "last" arg is given, dumps only the last file.

If "out=" file name specified, data will be also dumped to that file.

Examples:

```
cable_info filename  
cable_info file=filename  
cable_info last  
cable_info date=jun04  
cable_info past=15d out=cable_info.csv
```

Example for `cable_info` command:

```
cable_info /.../cables_1595764809124997.bin
```

```
time, source, timestamp, port, lid, guid, port_name, vendor, oui, pn, sn, rev  
nominalbitrate, cdrenabletxrx, inputeq, outputamp, outputtemp, fw_version  
rx_power.1.mw, rx_power.1.dbm, rx_power.2.mw, rx_power.2.dbm, rx_power  
rx_power.4.dbm, tx_bias.1, tx_bias.2, tx_bias.3, tx_bias.4, tx_power.1.r
```

```

2020-07-
26T15:00:12.742710, cable_info, 1595764812742710, 1, 117, 0x248a0703008f
hercules-01/U1/P1, Mellanox, 0x2c9, MC2207130-002, MT1442VS07035, A3, 2
m, Copper cable- unequalized, SDR/DDR/QDR/FDR, N/A, 1, 0, N/A
N/A, N/A, N/A, N/A, N/A, 5 8 11
0, OMA, 0.0, -999.999023438, 0.0, -999.999023438, 0.0, -999.999023438, 0.0
11-27, 8224, 0, 0x0, 0x0,
2020-07-
26T15:00:12.742710, cable_info, 1595764812742710, 1, 104, 0xe41d2d030010
e2edmz-02/U1/P1, Mellanox, 0x2c9, MC2207130-002, MT1442VS07035, A3, 2
m, Copper cable- unequalized, SDR/DDR/QDR/FDR, N/A, 1, 0, N/A
N/A, N/A, N/A, N/A, N/A, 5 8 11
0, OMA, 0.0, -999.999023438, 0.0, -999.999023438, 0.0, -999.999023438, 0.0
11-27, 8224, 0, 0x0, 0x0,
2020-07-
26T15:00:12.742710, cable_info, 1595764812742710, 3, 104, 0xe41d2d030010
e2edmz-02/U1/P3, Mellanox, 0x2c9, MC2207130-002, MT1411VS08914, A3, 2
m, Copper cable- unequalized, SDR/DDR/QDR/FDR, N/A, 1, 0, N/A
N/A, N/A, N/A, N/A, N/A, 5 8 11
0, OMA, 0.0, -999.999023438, 0.0, -999.999023438, 0.0, -999.999023438, 0.0
03-25, 8224, 0, 0x0, 0x0,
2020-07-
26T15:00:12.742710, cable_info, 1595764812742710, 1, 187, 0xe41d2d030050
forwarder/U1/P1, Mellanox, 0x2c9, MC2207130-002, MT1411VS08914, A3, 2
m, Copper cable- unequalized, SDR/DDR/QDR/FDR, N/A, 1, 0, N/A
N/A, N/A, N/A, N/A, N/A, 5 8 11
0, OMA, 0.0, -999.999023438, 0.0, -999.999023438, 0.0, -999.999023438, 0.0
03-25, 8224, 0, 0x0, 0x0,

```

Port Counters

The `port_counters` command is used to extract data in CSV format. It dumps counters matching a given text fragment or "counterset" for a date or range of dates.

Following is the help menu of `port_counters` command:


```
2,2020-07-
27T09:57:02,1595833022873374,0xb8599f0300355d6e,3,0,9881980474,0,4,
3,2020-07-
27T09:57:02,1595833022873396,0xb8599f0300355d6e,4,0,0,0,13394969590
4,2020-07-
27T09:57:02,1595833022873408,0xb8599f0300355d6e,9,24766362454,0,0,0
5,2020-07-
27T09:57:02,1595833022873419,0xb8599f0300355d6e,10,0,54725808986,0
```

Switch Temperature

The `switch_temperature` command is used to dump switch temperature info for a given date or range of dates into CSV files.

The following presents the help menu of the `switch_temperature` command:

```
CollectX: help switch_temperature
Usage:
    switch_temperature [TIME] [out=]
    [TIME] is one of the following:
                                last
                                date=

past=n[hours|days]
    [out=] is to specify output file (optional)
Description:
    Dump switch temperature info for a given date or
range of dates.
    If "out=" file name specified, data will be also
dumped to that file.
Examples:
    switch_temperature filename
    switch_temperature file=filename
    switch_temperature date=apr21
```



```
past=n[hours|days]
```

```
[out=] is to specify output file (optional)
```

Description:

Dump switch fans info for a given date or range of dates.

If "out=" file name specified, data will be also dumped to that file.

Examples:

```
switch_fans filename
```

```
switch_fans file=filename
```

```
switch_fans date=jun04
```

```
switch_fans past=15d out=switch_fans.csv
```

The following is an example of a `switch_fans` command run:

```
CollectX: switch_fans past=10m out=switch_fans.csv
```

```
time,source,timestamp,node_guid,sensor_index,fan_speed,
```

```
2020-10-
```

```
04T17:36:05.287397,0xe41d2d0300169e40,1601822165287397,0xe41d2d0300
```

```
2020-10-
```

```
04T17:36:05.287402,0xe41d2d0300169e40,1601822165287402,0xe41d2d0300
```

```
2020-10-
```

```
04T17:36:05.287403,0xe41d2d0300169e40,1601822165287403,0xe41d2d0300
```

```
2020-10-
```

```
04T17:36:05.287404,0xe41d2d0300169e40,1601822165287404,0xe41d2d0300
```

```
...
```

Switch General

The `switch_general` command is used to dump general switch info for a given date or range of dates into CSV files.

The following presents the help menu of `switch_general` command:

```
CollectX: help switch_general
```

```
Usage:
```

```
switch_general [TIME] [out=]
```

```
[TIME] is one of the following:
```

```
last
```

```
date=
```

```
past=n[hours|days]
```

```
[out=] is to specify output file (optional)
```

```
Description:
```

```
Dump switch general info for a given date or range of dates.
```

```
If "out=" file name specified, data will be also dumped to that file.
```

```
Examples:
```

```
switch_general filename
```

```
switch_general file=filename
```

```
switch_general date=jun04
```

```
switch_general past=15d out=switch_general.csv
```

The following is an example of a `switch_general` command run:

```
CollectX: switch_general past=10m out=switch_general.csv
```

```
time,source,timestamp,node_guid,serial_number,part_number,revision
```

```
2020-10-
```

```
25T11:41:05.183039,0xe41d2d0300169e40,1603618865183039,0xe41d2d0300
```

```
EB2F,A6,Scorpion IB
```

```
EDR,0,49152,7936,16383,1,1,19,255,255,0,32,0,49183,1,1,1,1,1,
```

```
2020-10-
```

```
25T11:42:05.559284,0xe41d2d0300169e40,1603618925559284,0xe41d2d0300
```

```
EB2F,A6,Scorpion IB
EDR,0,49152,7936,16383,1,1,19,255,255,0,32,0,49183,1,1,1,1,1,
2020-10-
...
```

Bare Metal - Bringup Mode – amBER Format

amBER is an output format designed for debugging a cluster in its bringup stage.

The following shows the help menu of the generate amBER report command:

```
CollectX: generate_amber_ib_csv past=1h out=amber_ib.csv
```

For example:

```
CollectX: help generate_amber_ib_csv
Usage:
    generate_amber_ib_csv TIME [report_type=] [out=]
[show_raw_data=]
    TIME can be specified as:
        date=
        past=n[hours|days] :
relative to the current time on the server.
        from= to=
    [out=] to specify output file
    [show_raw_data=t|f] boolean to show raw data as is.
Default: f
Description:
    Dump amBER IB report for a given date or range of
dates
Example:
    generate_amber_ib_csv past=10m
    generate_amber_ib_csv date=jul16 out=amber_ib.csv
```

TIME:

from='sep 23, 2021 16:05:00'

from='2021-09-23 16:05:00'

Fluent Bit Export

NVIDIA® UFM® Telemetry adds the ability to stream to multiple destinations using Fluent Bit. The streaming implementation can stream to any Fluent Bit export plugin, with the "Forward" plugin being particularly useful as it allows sending data to a customer-maintained Fluent Bit or FluentD instance which the customer can then configure as based on their requirements.

Exporting Data Using Fluent Bit Export

To export collected data from the UFM Telemetry docker image:

1. Load, configure, and run the docker image. See the details in the "[Software Management](#)" chapter.
2. Connect to "ufm-telemetry docker bash".

```
[root@r-ufm ~]# sudo docker exec -it ufm-telemetry bash
```

3. Configure/create export files `*.exp` in export directory `/config/fluent_bit_configs/` and set `enable=1` for plugins you want to run. Please see details in the "[Export Files](#)" section.
4. Enable Fluent Bit export by setting `plugin_env_FLUENT_BIT_EXPORT_ENABLE=1` in `/config/launch_ibdiagnet_config.ini`.

```
[root@r-ufm ~]# vi
/telemetry.config/launch_ibdiagnet_config.ini
...
[fluentbit_export]

plugin_env_FLUENT_BIT_EXPORT_ENABLE=1
plugin_env_FLUENT_BIT_CONFIG_DIR=/telemetry.config/fluent_bit_
```

```
plugin_env_LD_LIBRARY_PATH=/opt/mellanox/collectx/lib
...
```

Alternatively, you may do this using the configuration script `configure_ufm_telemetry_target.py` by running:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py
enable-streaming
```

This changes the value of the `plugin_env_FLUENT_BIT_EXPORT_ENABLE` parameter in the `launch_ibdiagnet_config.ini` file. See section "[Controlling Fluent Bit Streaming](#)" for more details.

5. Run destination programs that will receive data. See more details in the "[Data Forwarding](#)" section.
6. See the data on the receiving side.

lbdiagnet will collect and export data periodically as configured by `launch_ibdiagnet_config.ini` file using the `sample_rate` parameter.

Export Files

Export destinations are set by configuring `.exp` files or creating new ones. All export files are placed in the export configuration folder `/config/fluent_bit_configs`. The easiest way to start is to use documented example `exp`-files for the following plugins:

- forward
- stdout
- stdout_raw (this plugin is presented only in the Fluent Bit version installed in the UFM Telemetry docker image)

All plugins are disabled by default. To enable a plugin, set `enable=1`.

Export File Configuration Details

Each export destination has the following fields:

- name – configuration name
- plugin_name – Fluent Bit plugin name
- enable – 1 or 0 values to enable/disable this destination
- host – the host for Fluent Bit plugin
- port – port for Fluent Bit plugin
- msgpack_data_layout – the msgpacked data format. Default is `flb_std`. The other option is `custom`. See section "[Msgpack Data Layout](#)" for details.
- plugin_key=val – key-value pairs of Fluent Bit plugin parameter (optional)
- counterset/fieldset – file paths (optional). See the details in section "[Cset/Fset Filtering](#)".

Use "#" to comment line.

Msgpack Data Layout

Data layout can be configured using .exp files by setting "msgpack_data_layout=layout".

Two layouts are available:

1. "flb_std" data layout is an array of 2 fields: timestamp double value and a plain dictionary (key-value pairs). The standard layout is appropriate for all Fluent Bit plugins. For example:

```
[timestamp_val, {"timestamp"=>ts_val,  
type=>"counters/events", "source"=>"source_val",  
"key_1"=>val_1, "key_2"=>val_2, ...}]
```

2. "custom" data layout is a dictionary of meta-fields and counter fields. Values are placed into a separate plain dictionary. Custom data format can be dumped with

"stdout_raw" output plugin of fluent-bit installed or can be forwarded with "forward" output plugin.

Counters example:

```
{"timestamp"=>timestamp_val, "type"=>"counters",  
"source"=>"source_val", "values"=> {"key_1"=>val_1,  
"key_2"=>val_2, ...}}
```

Events example:

```
{"timestamp"=>timestamp_val, "type"=>"events",  
"type_name"=>"type_name_val", "source"=>" source_val",  
"values"=>{"key_1"=>val_1, "key_2"=>val_2, ...}}
```

Cset/Fset Filtering

Each export file can optionally use one cset and one fset file to filter UFM Telemetry counters and events data.

- Cset file contains tokens per line to filter data with `"type"="counters"`.
- Fset contains several blocks started with the header line `[event_type_name]` and tokens under that header. Fset file is used to filter data with `"type"="events"`.
 - Event type names can be prefixed to apply the same tokens to all fitting types. For example, to filter all ethtool events use `[ethtool_event_*]`.

If several tokens are needed to be matched simultaneously use "tok1+tok2+tok3". Exclusive tokens are available too: line "tok1+tok2-tok3-tok4" will filter names that match both tok1 and tok2 and do not match tok3 or tok4.

Both events and counters can be extended with aliased fields and new constant fields.

- “meta_field_aliases:exact_name=alias” will add new field/counter with name “alias_name” and copied value from the existing field/counter “exact_name”.
- “meta_field_add:new_name=constant_value” will add new field/counter with a name “new_name” and value “constant_value”

New fields should have unique names, otherwise, they will be ignored.

For more details see documentation in the files `ufm_enterprise.cset` and `ufm_enterprise.fset` under `/config/fluent_bit_configs`.

The following is the content of

`/config/fluent_bit_configs/ufm_enterprise.cset`:

```
# put tokens on separate lines

# Tokens are the actual name 'fragments' to be matched
#   port$           # match names ending with token "port"
#   ^port          # match names starting with token "port"
#   ^port$         # include name that is exact token "port"
#   port+xmit      # match names that contain both tokens "port"
# and "xmit"
#   port-support   # match names that contain the token "port"
# and do not match the "-" token "support"
#   -port          # exclude all names that contain the token
# "port"
#
# Tip: To disable counter export put a single token line that
# fits nothing

# Meta fields are user-defined additional fields of 2 types:
# aliases and new constant fields.
# - Aliases:
#   add data of field "exact_name" to meta fields of record
# with new "alias_name".
#   One field can have only one alias.
```

```

# Aliases match only exact names and will appear in data
record even if field is disabled by fset.
# Example:
# meta_field_alias:exact_name=alias_name
# - Constants:
# add new field "new_field_name" with constant data string
"constant_value" to the meta fields.
# Names should be unique.
# Example:
# meta_field_add:new_field_name=constant_value

# List of available counters:
#
#node_guid
#port_guid
#port_num
#lid
#link_down_counter
#link_error_recovery_counter
#symbol_error_counter
#port_rcv_remote_physical_errors
#port_rcv_errors
#port_xmit_discard
#port_rcv_switch_relay_errors
#excessive_buffer_errors
...

```

The following is the content of

```
/config/fluent_bit_configs/ufm_enterprise.fset:
```

```

# Put your events here

# Usage:
#

```

```

# [type_name_1]
# tokens
# [type_name_2]
# tokens
# [type_name_3]
# tokens
# ...

# Tokens are the actual name 'fragments' to be matched
#     port$           # match names ending with token "port"
#     ^port          # match names starting with token "port"
#     ^port$         # include name that is exact token "port"
#     port+xmit      # match names that contain both tokens "port"
and "xmit"
#     port-support   # match names that contain the token "port"
and do not match the "-" token "support"
#     -port          # exclude all names that contain the token
"port"

# Meta fields are user-defined additional fields of 2 types:
aliases and new constant fields.
# - Aliases:
#     add data of field "exact_name" to meta fields of record
with new "alias_name".
#     One field can have only one alias.
#     Aliases match only exact names and will appear in data
record even if field is disabled by fset.
#     Example:
#         meta_field_alias:exact_name=alias_name
# - Constants:
#     add new field "new_field_name" with constant data string
"constant_value" to the meta fields.
#     Names should be unique.
#     Example:
#         meta_field_add:new_field_name=constant_value

```

```
# The next example will export the whole "switch_fan" events and
events "CableInfo" filtered with token "port" :
# [switch_fan]
#
# [CableInfo]
# port

# To know which event type names are available use one of these
options:
#     1. Check export and find field
"type_name"=>"switch_temperature"
#         OR
#     2. Open log file "/tmp/ibd/ibdiagnet2_port_counters.log"
and find event types are printed to log:
#     ...
#     [info] type [CableInfo] is type of interest
#     [info] type [switch_temperature] is type of interest
#     [info] type [switch_fan] is type of interest
#     [info] type [switch_general] is type of interest
#     ...

# Corner cases:
# 1. Empty fset file will export all events.
# 2. Tokens written above/without [event_type] will be ignored.
# 3. If cannot open fset file, warning will be printed, all event
types will be exported.
```

Quick Start Guide for FluentD

1. Connect to a remote Linux machine via SSH and ensure docker is installed and started on it.

```
[root@r-ufm ~]# sudo service docker start
```

2. Pull FluentD image:

```
[root@r-ufm ~]# sudo docker pull fluentd
```

3. Create a configuration file for fluentd container.

```
[root@r-ufm ~]# export fluentd_dir=/tmp/fluentd
[root@r-ufm ~]# mkdir -p $ fluentd_dir
[root@r-ufm ~]# vim $ fluentd_dir/config.conf #fill it with
next configuration

<source>
  @type forward
  bind 0.0.0.0
  port 24432
</source>

<match ufm_telemetry>
  @type stdout
</match>
```

4. Start fluentd collector container.

```
[root@r-ufm ~]# sudo docker run -it --rm --network host -v
$fluentd_dir:/fluentd/etc fluentd -c /fluentd/etc/config.conf
-v
```

For more details refer to "[FluentD](#)" on docker hub.

Data Forwarding

1. Follow the instructions under "[Quick Start Guide for FluentD](#)" to prepare remote host with a running FluentD.
2. Follow the instructions under "[Exporting Data Using Fluent Bit Export](#)" to prepare UFM Telemetry with Fluent Bit export capability and ensure it matches the following configurations:
 - Fluent Bit is enabled (`plugin_env_FLUENT_BIT_EXPORT_ENABLE=1`) in the `launch_ibdiagnet_config.ini` file:

```
[root@r-ufm ~]# grep -a2 fluent
/config/launch_ibdiagnet_config.ini

[fluentbit_export]
plugin_env_FLUENT_BIT_EXPORT_ENABLE=1
plugin_env_FLUENT_BIT_CONFIG_DIR=/telemetry.config/fluent_
plugin_env_LD_LIBRARY_PATH=/opt/mellanox/collectx/lib
```

- Prepare a `forward.exp` file to send data to remote host where `fluentd` is running:

```
[root@r-ufm ~]# cat
/config/fluent_bit_configs/forward.exp

name=ufm-enterprise
enable=1
plugin_name=forward
host=10.209.36.248 # Remote host IP where fluentd is
running
port=24432
```

```
plugin_tag_match_pair=ufm_telemetry
```

3. Verify that data is streamed from the CollectX Telemetry plugin and is received on the FluentD collector.

UFM Telemetry Configuration Script

A script to facilitate the configuration of UFM Telemetry is located under the path `/config/configure_ufm_telemetry_target.py`.

The script is used to set and show sample rate duration, enable and disable streaming capabilities, add, remove, update, enable, disable and review target destinations to receive counters and cable info data, and import filters defined in files to filter streamed data.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py -h
usage: configure_ufm_telemetry_target.py <command> [<args>]
```

positional arguments:

```
{add-target, show-target, remove-target, enable-target, enable-
streaming, disable-target, disable-streaming, modify-target, import-
filter-file, disable-filter-file, set-sample-rate, show-sample-rate}
```

	Commands
add-target	Add a telemetry target
show-target	Show telemetry target(s)
remove-target	Remove a telemetry target
enable-target	Enable a telemetry target
enable-streaming	Enable telemetry streaming
disable-target	Disable a telemetry target
disable-streaming	Disable telemetry streaming
modify-target	Modify a telemetry target
import-filter-file	Import a telemetry target filter file
disable-filter-file	Disable telemetry target filter file
set-sample-rate	Set telemetry sample rate
show-sample-rate	Show telemetry sample rate

optional arguments:

-h, --help	show this help message and exit
-V, --version	Print version information

Controlling Fluent Bit Streaming

Fluent Bit data streaming is disabled by default. You may enable it by using the script argument `enable-streaming` (`disable-streaming` to disable). This changes the value of the `plugin_env_FLUENT_BIT_EXPORT_ENABLE` parameter in the `launch_ibdiagnet_config.ini` file.

```
[root@r-ufm ~]# grep plugin_env_FLUENT_BIT_EXPORT_ENABLE
/config/launch_ibdiagnet_config.ini
plugin_env_FLUENT_BIT_EXPORT_ENABLE=0
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py enable-
streaming
[root@r-ufm ~]# grep plugin_env_FLUENT_BIT_EXPORT_ENABLE
/config/launch_ibdiagnet_config.ini
plugin_env_FLUENT_BIT_EXPORT_ENABLE=1
```

Controlling Target Destinations

You can add, remove, update, enable, disable and review many target destinations to receive counters and cable info data.

Note

Use the flag `-h` to see the details of any operation.

Adding Destination Target

The parameter `add-target` adds and enables a destination target.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py add-
target -h
usage: configure_ufm_telemetry_target.py <command> [<args>] add-
target
    [-h] -n <[A-Za-z0-9_-] Name size: 32> -H <IPv4> -p <1-
65535> -m
    {extended,standard}

optional arguments:
  -h, --help            show this help message and exit
  -n <[A-Za-z0-9_-] Name size: 32>, --target-name <[A-Za-z0-9_-]
Name size: 32>
                        Target name
  -H <IPv4>, --target-host <IPv4>
                        IPv4 address
  -p <1-65535>, --target-port <1-65535>
                        Port number
  -m {extended,standard}, --target-message-type
{extended,standard}
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py add-
target --target-name ufm-telemetry --target-host 10.212.145.6 --
target-port 24453 -m standard
```

Displaying Destination Target Details

The parameter `show-target` displays the details of a destination target.

```
[root@r-ufm ~]#[root@r-ufm ~]#
/config/configure_ufm_telemetry_target.py add-target -h
usage: configure_ufm_telemetry_target.py <command> [<args>] add-
target
    [-h] -n <[A-Za-z0-9_-]> Name size: 32> -H <IPv4> -p <1-
65535> -m
    {extended,standard}

optional arguments:
  -h, --help            show this help message and exit
  -n <[A-Za-z0-9_-]> Name size: 32>, --target-name <[A-Za-z0-9_-]
Name size: 32>
                        Target name
  -H <IPv4>, --target-host <IPv4>
                        IPv4 address
  -p <1-65535>, --target-port <1-65535>
                        Port number
  -m {extended,standard}, --target-message-type
{extended,standard}
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py show-
target --target-name ufm-telemetry
Enabled:      Yes
  Name:      ufm-telemetry
  Enabled:   Yes
  Host:      10.212.145.6
  Port:      24453
  Message Type: Standard
```

Disabling Destination Target

The parameter `disable-target` disables a destination target.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py
disable-target -h
usage: configure_ufm_telemetry_target.py <command> [<args>]
disable-target
    [-h] -n TARGET_NAME

optional arguments:
  -h, --help            show this help message and exit
  -n TARGET_NAME, --target-name TARGET_NAME
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py
disable-target --target-name ufm-telemetry
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py show-
target --target-name ufm-telemetry
Enabled:      Yes
  Name:      ufm-telemetry
  Enabled:   No
  Host:      10.212.145.6
  Port:      24453
  Message Type: Standard
```

Enabling Destination Target

The parameter `enable-target` enables a destination target.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py enable-  
target -h  
usage: configure_ufm_telemetry_target.py <command> [<args>]  
enable-target  
    [-h] -n TARGET_NAME  
  
optional arguments:  
  -h, --help            show this help message and exit  
  -n TARGET_NAME, --target-name TARGET_NAME
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py enable-  
target --target-name ufm-telemetry  
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py show-  
target --target-name ufm-telemetry  
Enabled:      Yes  
Name:         ufm-telemetry  
Enabled:      Yes  
Host:         10.212.145.6  
Port:         24453  
Message Type: Standard
```

Modifying Destination Target

The parameter `modify-target` modifies a destination target.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py modify-  
target -h  
usage: configure_ufm_telemetry_target.py <command> [<args>]  
modify-target
```

```
[-h] -n TARGET_NAME [-H <IPv4>] [-p <1-65535>] [-m  
{extended, standard}]
```

optional arguments:

```
-h, --help                show this help message and exit  
-n TARGET_NAME, --target-name TARGET_NAME  
-H <IPv4>, --target-host <IPv4>  
                        IPv4 address  
-p <1-65535>, --target-port <1-65535>  
                        Port number  
-m {extended, standard}, --target-message-type  
{extended, standard}
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py modify-  
target --target-name ufm-telemetry --target-host 10.212.145.7 --  
target-port 24455 -m standard  
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py show-  
target --target-name ufm-telemetry  
Enabled:      Yes  
Name:         ufm-telemetry  
Enabled:      Yes  
Host:         10.212.145.7  
Port:         24455  
Message Type: Standard
```

Removing Destination Target

The parameter `remove-target` removes a destination target.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py remove-  
target -h  
usage: configure_ufm_telemetry_target.py <command> [<args>]  
remove-target  
    [-h] -n TARGET_NAME  
  
optional arguments:  
  -h, --help            show this help message and exit  
  -n TARGET_NAME, --target-name TARGET_NAME
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py remove-  
target --target-name ufm-telemetry  
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py show-  
target --target-name ufm-telemetry  
Enabled:      Yes  
Target ufm-telemetry is missing. Please add it first.
```

Data Filtration

The `configure_ufm_telemetry_target.py` script allows users to import filter files to enable filtering streamed data and to disable filter options.

Enabling Data Filtration

To enable filtration of the streamed counters and cable info data, users must create a file containing the appropriate RegEx patterns (one pattern per line to extract the required parameters data).

```
[root@r-ufm ~]# cat ~/counters_filter
```

```
lm_counter
Errors
```

Then they must import the filter file to a destination, specifying the type of data (counters or cable info) using the parameter `import-filter-file`.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py import-
filter-file -h
usage: configure_ufm_telemetry_target.py <command> [<args>]
import-filter-file
    [-h] -n TARGET_NAME -t {counters,fields} -f FILE_PATH

optional arguments:
  -h, --help            show this help message and exit
  -n TARGET_NAME, --target-name TARGET_NAME
  -t {counters,fields}, --target-filter-type {counters,fields}
  -f FILE_PATH, --file-path FILE_PATH
```

For example, to enable filtering streamed data and create filters:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py import-
filter-file --target-name ufm-telemetry --target-filter-type
counters --file-path ~/counters_filter
```

On the target destination side, users will receive all the counters include one of texts (lm_counter Errors).

Disabling Data Filtration

The parameter `disable-filter-file` disables an imported filtering file.

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py
disable-filter-file -h
usage: configure_ufm_telemetry_target.py <command> [<args>]
disable-filter-file
    [-h] -n TARGET_NAME -t {counters,fields}

optional arguments:
  -h, --help            show this help message and exit
  -n TARGET_NAME, --target-name TARGET_NAME
  -t {counters,fields}, --target-filter-type {counters,fields}
```

For example:

```
[root@r-ufm ~]# /config/configure_ufm_telemetry_target.py
disable-filter-file --target-name ufm-telemetry --target-filter-
type counters
```

On the target destination side, users will receive all the counters without filtering.

Settings and Configuration

Inside the container, the directory `/config` contains the configuration files for the NVIDIA® UFM® Telemetry application. The file `launch_ibdiagnet_config.ini` is the main configuration file.

The basic configurations of `launch_ibdiagnet_config.ini` are listed in the following table.

Section	Key	Type	Default Value	D
ibdiagnet	ibdiagnet_enabled	bool	true	E it
	data_dir	String	/data	D U is
	ibdiag_output_dir	String	/tmp/ibd	D it
	sample_rate	Int	-	Fi co co
	hca	String	mlx5_2	C p se ci a
	force_hca	bool	false	S
	app_name	String	/opt/collectx/bin/ibdiagnet	A fu it if
	topology_mode	String	discover	T
	topology_discovery_factor	Int	0	E d

Section	Key	Type	Default Value	D
				o fr
	m_key	int	-	S b c
xdr	XDR	bool	false	N o t a i
Retention	retention_enabled	bool	true	E r
	retention_interval	time	1d	l r b r
	retention_age	time	100d	P c
compression	compression_enable	bool	true	E c
	compression_interval	time	6h	l r b c
	compression_age	time	12h	P c
cable_info	cable_info_schedule	CSV	-	w T i

Log File Rotation

UFM telemetry log file “`ibdiagnet2_port_counters.log`” size is monitored by log rotation mechanism. This is highly relevant for cases of long execution time and/or high verbosity, where the number of logs can get excessively big.

To disable log rotation, verify that the following flag is set to 0 (default is 1):

```
plugin_env_CLX_LOG_ROTATE_ENABLED
```

To change the number of rotated files, set the following flag (default is 3):

```
plugin_env_CLX_LOG_ROTATE_NUM_FILES
```

To change the rotation's threshold, set the following flag (default is 100M), use [K|M|G] as units:

```
plugin_env_CLX_LOG_ROTATE_SIZE
```

There are three optional rotation methods, used in the following order:

1. `rotatelogs` - If this executable exists, it will be used for logs rotation, and the rotated files name will differ by index suffix.
2. `logrotate` - If this executable exists, it will be used for logs rotation, and the rotated files name will differ by timestamp suffix.
3. manual rotation - In case both executables are not available, UFM telemetry will manually rotate 2 log files. The older log file will have “.bck”

To skip options, the following flag set the executables to use (default is “rotatelogs,logrotate”):

```
plugin_env_CLX_LOG_ROTATE_APP
```

Auto Discovery Updates

The UFM Telemetry auto discovery feature enables in-process topology updates, eliminating the need for process restarts.

This is achieved through additional in-process ad-hoc discoveries and topology updates triggered by external events.

The following parameters configure the auto discovery updates feature:

Key	Type
<code>plugin_env_CLX_EXPORT_API_ENABLE_AD_HOC_DISCOVERY</code>	bool
<code>plugin_env_CLX_EXPORT_API_AD_HOC_DISCOVERY_WITH_IB_TRAP</code>	bool
<code>plugin_env_CLX_EXPORT_API_AD_HOC_DISCOVERY_WITH_MANAGER_IB_TRAP</code>	bool
<code>plugin_env_CLX_EXPORT_API_AD_HOC_DISCOVERY_MIN_INTERVAL_SECONDS</code>	int
<code>plugin_env_CLX_EXPORT_API_AD_HOC_DISCOVERY_WITH_CLX_RESTART</code>	bool
<code>plugin_env_CLX_EXPORT_API_AD_HOC_DISCOVERY_PERIODIC</code>	bool

Key	Type
plugin_env_CLX_EXPORT_API_IB_TRAP_LID	int

Port Counters

The table below outlines the supported configurations for enabling section collection.

To include a register for collection, certain configurations need to be added, some removed, and some may depend on other registers. Please refer to the table below for instructions:

Note

Please note that the 'arg' key names may differ.

Register / Page name	Configurations to be Applied
SLRIP	arg_slrip=--get_phy_info --enabled_regs slrip plugin_env_CLX_EXPORT_API_SKIP_SLRIP=0
SLRP	arg_slrp=--get_phy_info --enabled_regs slrp plugin_env_CLX_EXPORT_API_SKIP_SLRP=0
SLRG	arg_slrg=--get_phy_info --enabled_regs slrg

Register / Page name	Configurations to be Applied
	<code>plugin_env_CLX_EXPORT_API_SKIP_SLRG=0</code>
PTYS	<code>arg_ptys=--get_phy_info --enabled_regs ptys</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PTYS=0</code>
PPHCR	<code>arg_pphcr=--get_phy_info --enabled_regs pphcr</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PPHCR=0</code>
SLSIR	<code>arg_slsir=--get_phy_info --enabled_regs slsir</code> <code>plugin_env_CLX_EXPORT_API_SKIP_SLSIR=0</code>
SLTP	<code>arg_sltp=--get_phy_info --enabled_regs sltp</code> <code>plugin_env_CLX_EXPORT_API_SKIP_SLTP=0</code>
PMDR	<code>arg_pmdr=--get_phy_info --enabled_regs pmdr</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PMDR=0</code>
PPLL	<code>arg_ppll=--get_phy_info --enabled_regs ppll</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PPLL=0</code>
SLLM	<code>arg_sllm=--get_phy_info --enabled_regs sllm</code> <code>plugin_env_CLX_EXPORT_API_SKIP_SLLM=0</code>
PHY BER Params	<code>arg_phy_ber_params=--get_phy_info --enabled_regs ppbmp</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PHY_BER_PARAMS=0</code>
MRCS	<code>arg_mrccs=--get_phy_info --enabled_regs mrccs</code> <code>plugin_env_CLX_EXPORT_API_SKIP_MRCS=0</code>
ARHC	<code>arg_arhc=--get_phy_info --enabled_regs arhc</code> <code>plugin_env_CLX_EXPORT_API_SKIP_ARHC=0</code>
ARHR	<code>arg_arhr=--get_phy_info --enabled_regs arhr</code> <code>plugin_env_CLX_EXPORT_API_SKIP_ARHR=0</code>
PEMI Laser Source Advanced	<code>arg_pemi_laser_source_advanced=--get_phy_info --enabled_regs</code> <code>pemi_laser_source_module_advanced_samples</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PEMI_LASER_SOURCE_ADVANCED=0</code>
PEMI Laser Source Essential	<code>arg_pemi_laser_source_essential=--get_phy_info --enabled_regs</code> <code>pemi_laser_source_module_essential_s</code> <code>plugin_env_CLX_EXPORT_API_SKIP_PEMI_LASER_SOURCE_ESSENTIAL=0</code>

Register / Page name	Configurations to be Applied
PEMI Module Samples	<pre>arg_pemi_module_samples=--get_phy_info --enabled_regs pemi_module_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_MODULE_SAMPLES=0</pre>
PEMI Module Status Properties	<pre>arg_pemi_module_status_properties=--get_phy_info --enabled_regs pemi_module_status_properties plugin_env_CLX_EXPORT_API_SKIP_PEMI_MODULE_STATUS_PROPERTIES=0</pre>
Performance Histogram Ports Control	<pre>plugin_env_CLX_EXPORT_API_SKIP_PERFORMANCE_HISTOGRAM_PORTS_CONTROL=0</pre>
Performance Histogram Ports Data	<pre>plugin_env_CLX_EXPORT_API_SKIP_PERFORMANCE_HISTOGRAM_PORTS_DATA=0</pre>
PEMI SNR Samples	<pre>arg_pemi_snr_samples=--get_phy_info --enabled_regs pemi_snr_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_SNR_SAMPLES=0</pre>
PEMI Laser Samples	<pre>arg_pemi_laser_samples=--get_phy_info --enabled_regs pemi_laser_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_LASER_SAMPLES=0</pre>
PEMI Pam4 Samples	<pre>arg_pemi_pam4_samples=--get_phy_info --enabled_regs pemi_pam4_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_PAM4_SAMPLES=0</pre>
PEMI Pre FEC BER Samples	<pre>arg_pemi_pre_fec_ber_samples=--get_phy_info --enabled_regs pemi_pre_fec_ber_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_PRE_FEC_BER_SAMPLES=0</pre>
PEMI Pre FEC BER Properties	<pre>arg_pemi_pre_fec_ber_properties=--get_phy_info --enabled_regs pemi_pre_fec_ber_properties plugin_env_CLX_EXPORT_API_SKIP_PEMI_PRE_FEC_BER_PROPERTIES=0</pre>
PEMI Ferc Samples	<pre>arg_pemi_ferc_samples=--get_phy_info --enabled_regs pemi_ferc_samples plugin_env_CLX_EXPORT_API_SKIP_PEMI_FERC_SAMPLES=0</pre>
PHY Counters	<pre>arg_phy_cntrs=--get_phy_info --enabled_regs dd_ppcnt_plc plugin_env_CLX_EXPORT_API_SKIP_PHY_CNTRS=0</pre>
PHY Statistics	<pre>arg_phy_stat=--get_phy_info --enabled_regs dd_ppcnt_plsc plugin_env_CLX_EXPORT_API_SKIP_PHY_STAT=0</pre>

Register / Page name	Configurations to be Applied
PHY InfiniBand General Counters	<pre>arg_phy_ib_general_cntrs=--get_phy_info --enabled_regs dd_ppcnt_gen_counters plugin_env_CLX_EXPORT_API_SKIP_PHY_IB_GENERAL_CNTRS=0</pre>
Histograms	<pre>arg_hist=--get_phy_info --enabled_regs dd_ppcnt_rsfc plugin_env_CLX_EXPORT_API_SKIP_HIST=0</pre>
Troubleshoot	<pre>arg_troubleshoot=--get_phy_info --enabled_regs dd_pddr_ti plugin_env_CLX_EXPORT_API_SKIP_TROUBLESHOOT=0</pre>
Operation Info	<pre>arg_operation_info=--get_phy_info --enabled_regs dd_pddr_ plugin_env_CLX_EXPORT_API_SKIP_OPERATION_INFO=0</pre>
Link Down Info	<pre>arg_link_down_info=--get_phy_info --enabled_regs dd_pddr_ plugin_env_CLX_EXPORT_API_SKIP_LINK_DOWN_INFO=0</pre>
Link Up Info	<pre>arg_link_up_info=--get_phy_info --enabled_regs dd_pddr_lu plugin_env_CLX_EXPORT_API_SKIP_LINK_UP_INFO=0</pre>
PLR	<pre>arg_plr=--get_phy_info --enabled_regs dd_ppcnt_plr plugin_env_CLX_EXPORT_API_SKIP_PLR=0</pre>
Port VL	<pre>arg_port_vl=--per_slvl_cntrs plugin_env_CLX_EXPORT_API_SKIP_PORT_VL=0</pre>
Congestion Control Port VL	<pre>arg_congestion_control=--congestion_counters plugin_env_CLX_EXPORT_API_SKIP_CC_PORT_VL=0</pre>
MLNX Counters Page0	<pre>arg_mlnx_counters_page0=--sc plugin_env_CLX_EXPORT_API_SKIP_MLNX_COUNTERS_PAGE0=0</pre>
MLNX Counters Page1	<pre>arg_mlnx_counters_page1=--sc plugin_env_CLX_EXPORT_API_SKIP_MLNX_COUNTERS_PAGE1=0</pre>
MLNX Counters Page255	<pre>arg_mlnx_counters_page255=--sc plugin_env_CLX_EXPORT_API_SKIP_MLNX_COUNTERS_PAGE255=0</pre>
Sharp PM Counters	<pre>arg_sharp_pm_counters=--sharp --sharp_opt dsc</pre>

Register / Page name	Configurations to be Applied
	<code>plugin_env_CLX_EXPORT_API_SKIP_SHARP_PM_COUNTERS=0</code>
General Info	<code>plugin_env_CLX_EXPORT_API_SKIP_GENERAL_INFO=0</code>
Device Temperature	<code>plugin_env_CLX_EXPORT_API_SKIP_DEV_TEMP=0</code>
Port Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_COUNTERS=0</code>
Port Counters Extended	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_COUNTERS_EXTENDED=0</code>
Extended Speeds Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_EXTENDED_SPEEDS_COUNTERS=0</code>
LLR Statistics	<code>plugin_env_CLX_EXPORT_API_SKIP_LLR_STATISTICS=0</code>
Port Rcv Error Details	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_RCV_ERROR_DETAILS=0</code>
Port Xmit Discard Details	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_XMIT_DISCARD_DETAILS=</code>
Port RN Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_RN_COUNTERS=0</code>
Port HBF Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_HBF_COUNTERS=0</code>
Fast Recovery Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_FAST_RECOVERY=0</code>
Recovery Policy Type Counters	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_RECOVERY_POLICY_COUNTERS=</code>
Recovery Policy Type Config	<code>plugin_env_CLX_EXPORT_API_SKIP_PORT_RECOVERY_POLICY_CONFIG=0</code>

Register / Page name	Configurations to be Applied
PPCNT Recovery Counters	arg_ppcnt_recovery_counters=--get_phy_info --enabled_regs ppcnt_plrc plugin_env_CLX_EXPORT_API_SKIP_PPCNT_RECOVERY_COUNTERS=0
Port General Counters	plugin_env_CLX_EXPORT_API_SKIP_PORT_GENERAL_COUNTERS=0
Port Hierarchy	plugin_env_CLX_EXPORT_API_SKIP_PORT_HIERARCHY=0
Link Partner	plugin_env_CLX_EXPORT_API_SKIP_LINK_PARTNER_EXT=0
Ports	plugin_env_CLX_EXPORT_API_SKIP_PORTS=0
Calculated Info	plugin_env_CLX_EXPORT_API_SKIP_CALC_INFO=0
Mad Statistics	arg_mad_stats=--mad_stat plugin_env_CLX_EXPORT_API_SKIP_MAD_STATS=0
Performance Histogram Info	plugin_env_CLX_EXPORT_API_SKIP_PERFORMANCE_HISTOGRAM_INFO
Performance Histogram Buffer Control	plugin_env_CLX_EXPORT_API_SKIP_PERFORMANCE_HISTOGRAM_BUFFER CLX_EXPORT_API_PERFORMANCE_HISTOGRAM_COLLECT_ALL=1
Performance Histogram Buffer Data	plugin_env_CLX_EXPORT_API_SKIP_PERFORMANCE_HISTOGRAM_BUFFER CLX_EXPORT_API_PERFORMANCE_HISTOGRAM_COLLECT_ALL=1

Events

The info below outlines the supported configurations for enabling section collection.

Enable PPCC

```
plugin_env_CLX_EXPORT_API_DISABLE_PPCCINFO
```

The following events are created:

```
ppcc_algo_config, ppcc_algo_config_params, ppcc_algo_config_support,  
ppcc_algo_counters
```

Switch Power Sensors Data

To enable Switch power sensors, ensure that the following line is added and not commented:

```
arg_x= --get_phy_info --enabled_reg mvcr # x should be replaced  
with the next available index!
```

Verify the following line does not exist / is set to 0:

```
plugin_env_CLX_EXPORT_API_DISABLE_SWITCHINFO=0
```

Switch Power Supplies Data

To enable switch power supplies, ensure that the following line is added and not commented:

```
arg_x= --get_phy_info --enabled_reg msps # x should be replaced  
with the next available index!
```

Verify the following line does not exist / is set to 0:

```
plugin_env_CLX_EXPORT_API_DISABLE_SWITCHINFO=0
```

Managed Switch Data Collection

Prerequisite: Access to UFM that is running the sysinfo plugin. The following configs are mandatory to enable the collection.

To enable the feature, run:

```
plugin_env_CLX_EXPORT_API_DISABLE_MANAGED_SWITCHINFO=0
```

UFM endpoint:

```
plugin_env_MANAGED_SWITCH_DATA_EP=https://localhost/ufmRest/plugin/sysinfo/query
```

UFM token:

```
plugin_env_CLX_UFM_TOKEN=YWRtaW46MTIzNDU2
```

The UFM Telemetry server endpoint must be the same as the `PROMETHEUS_ENDPOINT`

```
plugin_env_CLX_EXPORT_API_MANAGED_SWITCH_CB_EP=http://localhost:1234/manag
```

The following configs are optional:

- The list of managed switches to sample, the default are all the managed switches on the fabric, defined by the sysinfo plugin:

```
plugin_env_CLX_EXPORT_API_MANAGED_SWITCH_LIST=11.222.33.44, 11.333.44
```

- `sample_rate` of `managed_switches(seconds)` should not be set faster than switch collection sample rate, default is 10 minutes.

```
plugin_env_CLX_EXPORT_API_MANAGED_SWITCH_INTERVAL=600
```

NVLink Info

The following configuration should be applied:

*the arg key name may differ

```
arg_nvlink_info=--nvlink  
plugin_env_CLX_EXPORT_API_SKIP_NVLINKINFO=0
```

Prometheus Endpoint Support

Prometheus Endpoint

UFM Telemetry can expose an http or https endpoint to allow simple and effective integration with monitoring systems that work in poll mode and support Prometheus, CSV, or JSON data formats. The endpoint provides only the last data sample. The user cannot obtain statistics for time points in the past.

Supported Formats

An http endpoint provides data in Prometheus format by default. It also supports JSON and CSV formats. The user can request the desired format using a URL prefix, as shown in the table below.

Data Format	URL Prefix
Prometheus	-
JSON	/json
CSV	/csv

Data Filtering

An http endpoint can provide all sampled data using the default `/metrics` URL. The filtering functionality described in the [Cset/Fset Filtering](#) section is also supported. To use it place `<name>.cset` or `<name>.fset` file in appropriate folders. This folder should be stated in configuration file. See section "[Configuring Data Polling Endpoint](#)" for more details.

The Extended counter set filtering, as described below, presents an alternative approach to filtering functionality by enabling counters and field selection.

A filter file name is included in the URL to request that the data be filtered through the particular `.cset` / `.fset` / `.xcset` file the user intends. For example, if there are two

filter files named `name1.cset` and `name2.cset`, then URLs `/name1` (or `/cset/name1`) and `/name2` (or `/cset/name2`) can be used to get filtered output described in these files accordingly.

The URL prefixes `/cset`, `/fset` and `/xcset` can also be used to specify which filter file is meant.

URL	File Extension	Folder Parameter in Configuration File	Note
<code>/cset</code>	<code>*.cset</code>	<code>plugin_env_PROMETHEUS_CSET_DIR</code>	If the <code>cset</code> folder is not explicitly specified in the configuration file, then the <code>cset</code> directory is set the same as the <code>fset</code> directory.
<code>/fset</code>	<code>*.fset</code>	<code>plugin_env_PROMETHEUS_FSET_DIR</code>	If the <code>fset</code> folder is not explicitly specified in the configuration file, then the <code>fset</code> directory is set the same as the <code>cset</code> directory.
<code>/xcset</code>	<code>*.xcset</code>	<code>plugin_env_PROMETHEUS_XCSET_DIR</code>	If the <code>xcset</code> folder is not explicitly specified in the configuration file, then the <code>xcset</code> directory is set the same as the <code>fset</code> directory.

Note

If a URL prefix is not specified, then the filter file will be searched under both `cset` and `fset` folders. If they both have files with the same names, then both filters will be applied.

Extended Counter Set Filtering

The http server provides an optional Extended counter set (`xcset`) selection mechanism in addition to the counter set (`cset`) and field set (`fset`) filtering. The Extended Counterset allows the user to generate an output record which contains data from both 'counters' and 'event' data records with the same index, which in the context of UFM Telemetry is generally the `guid/port_num`. To define an extended counter set, a file or group of files with the `.xcset` extension must be placed in its designated directory or adjacent to existing field or counter sets.

Each line of the file may contain:

- Selection of a counter with an optional alias in the format “`counter[=alias]`”
- Selection of a type's field with an optional alias in the format “`type.field[=alias]`”
- Reference to another file to be included “`file.xcset`”

Extended counter set files are searched for in the same directory as the source `xcset`.

Aliases are not mandatory, but if provided, they are used to name the selected counter or field in the output. Empty lines and comments that begin with the “#” sign are disregarded.

Rendering Hints

Extended counter sets support rendering hints to adjust attribution and representation of the metrics values.

The hints are given as a coma-separated list of key=value pairs following the field selection line after the semicolon (`;`) character.

```
counter[=alias];key[=value][,key[=value]]*
```

For Example:

```
port_guid;label,hex,default=undefined
hw_port_state;lookup=printable_port_states
```

Supported rendering hints are the following:

		Description	
hex	n/a	Requests a numeric value to be rendered hexadecimal	<code>port_num;hex</code>
label	n/a	Attributes the field as Prometheus label	<code>host_name;label</code>
default	value	Sets a string value to be rendered in case of data for the field is missing	<code>temperature;default=unknown</code>
const	value	Add the marked field as constant value to the output	<code>context;const=oberon</code>
lookup	name	Use the named lookup table to replace the value when rendering.	<code>hw_port_state;lookup=printable_port_states</code>

URL Prefixes Priority

URL prefixes can be used to manipulate data output. It is important to use the prefixes in the correct order as they have assigned priorities. The table below shows URL prefixes priority assignments with examples:

Priority	Prefix	Link Examples	Description
1	/labels	/labels/metrics, /metrics	Used to show labels from metadata files
2	/json, /csv	/json/metrics, /csv/metrics, /labels/json/metrics, /labels/csv/metrics	Used to specify output format
3	/cset, /fset, /xcset	/cset/filter1, /fset/filter2, /labels/cset/filter1, /labels/fset/filter2, /json/cset/filter1, /json/fset/filter2, /csv/cset/filter1, /csv/fset/filter2, /csv/xcset/ib, /labels/json/cset/filter1, /labels/json/fset/filter2, /labels/csv/cset/filter1, /labels/csv/fset/filter2	Used to specify which type of filter file should be applied

Sharding Data Requests

The shard data filter is useful when metric scraping loads need to be distributed over time or across consumer space. Below are several examples of how to use sharding in queries.

1. Shard counters and events by `node_guid`, serialize to CSV:

```
curl -v http://0.0.0.0:9352/csv/metrics?
num_shards=2&shard=0&sharding_field=node_guid
```

2. Shard by `node_guid` and filter by port number:

```
curl -v http://0.0.0.0:9352/csv/metrics?
num_shards=2&shard=0&sharding_field=node_guid&port_num_eq_1
```

3. Select a specific counter set explicitly:

```
curl -v http://0.0.0.0:9352/csv/cset/minimal?
num_shards=2&shard=0&sharding_field=node_guid
```

4. Use a predefined fieldset and shard by port number (using the port field):

```
curl -v http://0.0.0.0:9352/csv/fset/low_freq?
num_shards=2&shard=0&sharding_field=port
```

Configurable Indication of Missing Data

By default, numeric fields with no data are set to `0`. This behavior can be modified to show `'N/A'` instead.

Flag	Default Value	Effect
<code>NA_INDICATION</code>	<code>false</code>	Sets missing field values to <code>'N/A'</code> instead of <code>0</code>
<code>NA_INDICATION?</code> <code>plugin_env_CLX_CLEAN_DATA_EVERY_ITER</code>	<code>false</code>	Resets field values on each data collection iteration

Configuring Data Polling Endpoint

To configure the Prometheus endpoint, the keys listed below need to be set in the `launch_ibdiagnet_config.ini` file.

```
plugin_env_PROMETHEUS_ENDPOINT
http://0.0.0.0:9100
plugin_env_PROMETHEUS_PROXY_ENDPOINT_PORT    9200
plugin_env_PROMETHEUS_INDEXES                port_num
plugin_env_PROMETHEUS_FSET_INDEXES           port,lid,guid,
[CableInfo]^port_guid,^Port$
plugin_env_PROMETHEUS_CSET_DIR
/config/prometheus_configs/cset
...
```

There are several options related to configuring the HTTP polling endpoint. The key `plugin_env_PROMETHEUS_ENDPOINT` is used to configure the IP interface for endpoint binding. The “0.0.0.0” part in the setting above means that any of the host’s valid IP addresses can be used. Note that the user can also specify the host’s IP address explicitly.

The `plugin_env_PROMETHEUS_ENDPOINT` key also configures the data transport. For regular HTTP, prefix to `http`. To send over a TLS connection, set the prefix to `https`, set the above mandatory parameters (keys), and select the existing security keys as follows.

A DH (key exchange protoon) file can also be specified if needed as follows:

```
plugin_env_CLX_SSL_DH_FILE=/certs/dh.pem
```

To use custom labels for Prometheus statistics, a metadata file is used. For details about labels and label file format, see sections "[Prometheus Labels](#)" and "[Prometheus Label Generation](#)".

There are several options that allow configuring metadata. The file containing the labels used in Prometheus generation is set as follows:

```
plugin_env_CLX_METADATA_FILE=/config/labels.txt
```

The user can create the metadata file upon system setup or use a script to generate it automatically via script, using the following parameter:

```
plugin_env_CLX_METADATA_COMMAND=/opt/mellanox/collectx/telem/bin/gc
--fabric compute --file /var/log/ibdiagnet2.ibnetdiscover --
output /config/labels.txt
```

In the above example, the script generates metadata from `/var/log/ibdiagnet2.ibnetdiscover`. If the user wishes to create the label file manually, the above option should be commented out to prevent periodic overwriting of the content of the metadata file.

By default, the Prometheus endpoint provides statistics with the collection timestamps. The user can decide whether counter values will be passed with or without timestamps by setting the `plugin_env_PROMETHEUS_SHOW_TIMESTAMPS` parameter to T (true) or F (false), respectively. For example, to send counter values without timestamps, set the parameter as follows:

```
plugin_env_PROMETHEUS_SHOW_TIMESTAMPS=F
```

To use data filters folders with counter set, field sets, and extended counter sets, the directories where the files are stored should be configured as follows:

```
plugin_env_PROMETHEUS_CSET_DIR=/telemetry.config/prometheus_configs
plugin_env_PROMETHEUS_FSET_DIR=/telemetry.config/prometheus_configs
plugin_env_PROMETHEUS_XCSET_DIR=/telemetry.config/prometheus_configs
```

Warning

Any parameters not explicitly documented should not be changed and should be considered read-only.

Prometheus Labels

For use cases such as UFM Enterprise or UFM Cyber AI where the network topology is known, a human-readable name can be presented based on the GUID.

```
# TYPE PortXmitDataExtended counter
# TYPE PortXmitPktsExtended counter
PortXmitDataExtended{source="0x0002c90300f172a0",
node_guid="2c90300f172a0", port_guid="2c90300f172a2",
port_num="2"} 85554128244 1628683905941
PortXmitPktsExtended{source="0x0002c90300f172a0",
node_guid="2c90300f172a0", port_guid="2c90300f172a2",
port_num="2"} 1188251785 1628683905941
```

For integration with third-party applications, labels which are more human-readable may be generated using a labels metadata file, as described below.

Prometheus Label Generation

To generate custom labels, a file containing key-value pairs is used. When the keys are matched, the key-value pairs added to the Prometheus labels are generated.

The following is an example of the format of a labels metadata file:

```
ec0d9a0300b41a50_36|port_id|ec0d9a0300b41a50_36|device_name|Switch:
Mellanox
Technologies|device_type|switch|fabric|compute|hostname||node_desc
ec0d9a0300b41a50_37|port_id|ec0d9a0300b41a50_37|device_name|Switch:
Mellanox
Technologies|device_type|switch|fabric|compute|hostname||node_desc
ec0d9a0300b41a58_1|port_id|ec0d9a0300b41a58_1|device_name||device_
node|level||peer_level|leaf
98039b0300640b92_1|port_id|98039b0300640b92_1|device_name||device_
1|node_desc|agx-1 mlx5_0|level|server|peer_level|leaf
```

```

98039b0300640c22_1|port_id|98039b0300640c22_1|device_name||device_
2|node_desc|agx-2 mlx5_0|level|server|peer_level|leaf
0002c90300f172a0_2|port_id|0002c90300f172a0_2|device_name||device_
3|node_desc|agx-3 mlx4_0|level|server|peer_level|leaf
98039b0300640b9a_1|port_id|98039b0300640b9a_1|device_name||device_
3|node_desc|agx-3 mlx5_0|level|server|peer_level|leaf

```

The following is an example of the generated Prometheus output:

```

# TYPE infiniband_port_xmit_data_bytes counter
# TYPE infiniband_port_rcv_data_bytes counter
# TYPE infiniband_link_error_recovery_events counter
# TYPE infiniband_link_downed_events counter
# TYPE infiniband_cbw gauge
infiniband_port_xmit_data_bytes
{port_id="0002c90300f172a0_2", ADDITIONAL_LABELS} 82218360540
1628602711924
infiniband_port_rcv_data_bytes
{port_id="0002c90300f172a0_2", ADDITIONAL_LABELS} 82218429458
1628602711924
infiniband_link_error_recovery_events
{port_id="0002c90300f172a0_2", ADDITIONAL_LABELS} 0
1628602711924
infiniband_link_downed_events
{port_id="0002c90300f172a0_2", ADDITIONAL_LABELS} 0
1628602711924
infiniband_cbw
{port_id="0002c90300f172a0_2", ADDITIONAL_LABELS}} 0
1628602711924

where ADDITIONAL_LABELS include:
    hostname="agx-3"
    node_desc="agx-3 mlx5_0"
    device_name=""

```

```
device_type="host"  
fabric="compute"  
level="server"  
peer_level="leaf"
```

To enable this functionality, the following additional keys need to be configured:

```
plugin_env_CLX_EXPORT_API_IBNETDISCOVER_RUN_ONCE 1    # Without  
this, the gen_metadata.py script cannot generate the human  
readable names, nor the level and peer_level.  
plugin_env_CLX_METADATA_FILE /path/to/labels/file  
plugin_env_CLX_METADATA_COMMAND "python3  
/opt/mellanox/collectx/telem/bin/gen_metadata.py --fabric  
compute --file /var/log/ibdiagnet2.ibnetdiscover -o  
/path/to/labels/file"
```

To test, the `curl` command can be used as follows:

```
[root@jazz11 /]# curl --silent IP_ADDR_OF_HOST:9100/metrics  
|egrep "xmit|rcv" | tail  
port_xmit_discard{device_name="",device_type="host",fabric="compute  
mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"} 0  
1629194120043  
port_rcv_switch_relay_errors{device_name="",device_type="host",fab  
mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"} 0  
1629194120043  
port_rcv_constraint_errors{device_name="",device_type="host",fabric  
mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"} 0  
1629194120043
```

```
port_xmit_constraint_errors{device_name="",device_type="host",fabr:
mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"} 0
1629194120043
```

Distributed Telemetry - Switch Telemetry Agent

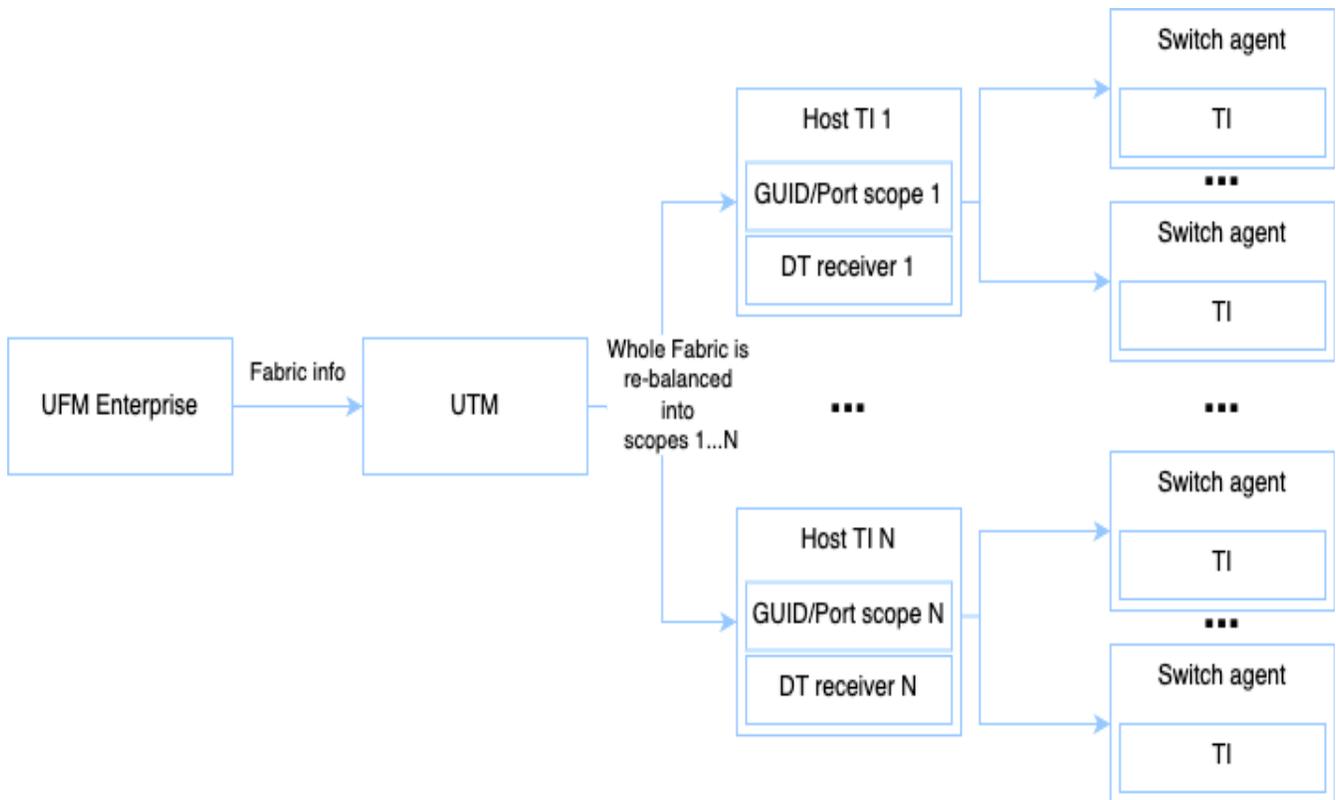
Overview

Distributed Telemetry (DT) is UFM Telemetry mode when the whole fabric telemetry is sampled from managed switches and hosts.

- Managed switch samples itself and hosts connected directly to it
- Each managed switch TI reports to one of several host TIs via MADs
- If a fabric GUID/Port can be sampled, but not sampled by a switch, host TI will sample it.

The whole process is orchestrated by [UFM Telemetry Manager \(UTM\)](#) on the top load balancer.

Switch telemetry is organized as a docker container (Switch Agent) with a telemetry package inside.



Distributed Telemetry components relation

Terminology

Switch Telemetry docker image contains:

- Switch Agent is an HTTP server running inside of the container.
- Switch Telemetry Instance (Switch TI) that can be started or stopped within the container

To deploy/remove Switch Agent = to deploy/remove Switch Telemetry Container.

Deployment

The deployment process is described in [UFM Telemetry Manager \(UTM\)](#).

Note

Switch Telemetry docker image is being docker pulled by the deployment script.

After preparing the setup enable Distributed Telemetry in `utm_config.ini`, as explained in the chapter `Configuration File`.

Running Distributed telemetry via HTTP API

For the sake of simplicity, the commands are listed here in UTM standalone mode only.

Info

UTM HTTP API access depends on the UTM run mode. Please refer to the section `REST API` of [UFM Telemetry Manager \(UTM\)](#) for more details.

To get help for all HTTP API endpoints use `/help` endpoint:

```
curl -sk https://127.0.0.1:8888/help
```

UTM HTTP API allows users to:

- get the status of Switch Agent/Telemetry instances
- deploy/remove Switch Agent containers
- start/stop Switch Telemetry inside of deployed containers
- set switches IP list to be periodically monitored.

Recommended flow to work with Distributed Telemetry:

- Deploy:

- Check the switches status and find IP list to work with
- Set monitoring switch list and deploy Switch Agents to this list
- Start switch TIs
- Cleanup:
- Stop switch TIs
- Remove Switch Agents
- Check switches status

Detailed instructions for Switch Agent and Switch Telemetry are listed in the following subsections.

API for Switch Agents

- Get the status of the managed switches in JSON format.

```
# all managed switches
curl -sk https://127.0.0.1:8888/managed_switches_status

# managed switches set to periodic monitoring only:
curl -sk https://127.0.0.1:8888/managed_switches_status?
monitored_only=1
```

`/status` endpoint provides JSON object per managed switch, which shows basic info about the switch, status of Switch Agent, and Switch Telemetry (If Switch Agent is installed to the switch).

- In the case of non-default switch credentials, upload them into UTM.

```
curl -s http://127.0.0.1:8888/set_switch_creds?ip=
{SWITCH_IP}&user={SWITCH_USER}&pass={SWITCH_PASSWORD}"
```

- Set switch IP list for periodic monitoring. Monitoring updates switch information for `/managed_switches_status` endpoint.

```
# monitor only IP1,IP2:
curl -sk https://127.0.0.1:8888/switch_mon_list?
ip_list=IP1,IP2

# monitor all managed switches
curl -sk https://127.0.0.1:8888/switch_mon_list?ip_list=all
```

- Deploy switch agents to a list of managed switches.

```
# deploy to all the managed switches:
curl -sk https://127.0.0.1:8888/deploy_switch_agents?
ip_list=all

# deploy to switches with IPs IP1 and IP2:
curl -sk https://127.0.0.1:8888/deploy_switch_agents?
ip_list=IP1,IP2
```

- Remove switch agents from a list of managed switches

```
# deploy to all the managed switches:
curl -sk https://127.0.0.1:8888/remove_switch_agents?
ip_list=all

# deploy to switches with IPs IP1 and IP2:
curl -sk https://127.0.0.1:8888/remove_switch_agents?
ip_list=IP1,IP2
```

API for Switch Telemetry:

- Start switch telemetry. Note at least one managed host TI should run.

```
# for all running Switch Agents
curl -sk https://127.0.0.1:8888/start_switch_telemetry

# at a specific switch IP:
curl -sk https://127.0.0.1:8888/start_switch_telemetry?ip=IP1
```

- Stop switch telemetry:

```
# for all running Switch Agents
curl -sk https://127.0.0.1:8888/stop_switch_telemetry

at a specific switch IP:
curl -sk https://127.0.0.1:8888/stop_switch_telemetry?ip=IP1
```

Supported Telemetry Fields

Ports Counters Info

The table below lists the names of Registers/Pages supported by telemetry, along with their respective field lists.

Register / Page Name	Field List
SLRIP	ffe_tap[0-8]_ib[0-3]_lane[0-3] ffe_tap_en_ib[0-3]_lane[0-3]mixer_offset[0-1]_ib[0-3]_lane[0-3]saved[0-1]
SLRP	crnt_bgn_offset_n[0-1]_lane[0-3] crnt_bgn_offset_p[0-1]_lane[0-3]dp_sel_lane[0-3]gctrl_bin_bgn[0-1]_n_lan
SLRG	Lane[0-3]Grade dn_eye_grade_lane[0-3]fom_ext_conf_cap_lane[0-3]fom_ext_conf_lane[0-
PTYS	data_rate_oper
PPHCR	active_hist_type bin_range_[0-15]_high_valbin_range_[0-15]_low_valbin_range_write_mask
SLSIR	ae_state_lane[0-3] cal_abort_cnt_lane[0-3]cal_done_cnt_lane[0-3]cdr_abort_cnt_lane[0-3]cdr
SLTP	alev_minus_bfm2_lane[0-3] alev_plus_bfm2_lane[0-3]blev_lane[0-3]lower_eye_amp_lane[0-3]main_tap
PMDR	clp_[1,4]x cluster_rx_lane[0-3]cluster_tx_lane[0-3]gb_dp_numgb_validgearbox_die_r
PPLL	ae_pll[0-3] analog_var_pll[0-3]bg_trimbg_trim_pll[0-24]bg_trim_validbg_trim_valid_pl 3]num_plls_[5,7,16]nmppll_lockdet_statepll_pwrup_pll[0-3]pll_speed_pll[0-3]
SLLM	ber_mon_exp_lane[0-3] ber_mon_lane[0-3]ber_mon_mantissa_lane[0-3]ctle_peq_cnt_lane[0-3]ctle
PHY BER Params	alarm_th_eff_ber alarm_th_raw_beralarm_th_sym_bernormal_th_eff_bernormal_th_raw_ber
MRCS	max_delta_freq_[0-1]

Register / Page Name	Field List
	max_freq_[0-1]measured_freq_[0-1]min_freq_[0-1]
ARHC	arhc_bin[0-1]_width arhc_bin_numarhc_bin_num_maxarhc_bin_width_modearhc_clear_on_rea
ARHR	arhr_bin_num arhr_histogram_bin_[0-15]
PEMI Module Samples	pemi_dp_st_lane[0-7] pemi_module_stpemi_rx_power_lane[0-7]pemi_tx_bias_lane[0-7]pemi_tx_l
PEMI SNR Samples	snr_host_lane[0-7] snr_media_lane[0-7]
PEMI Laser Samples	laser_frequency_error_lane[0-7] els_cooled_laser_temperature_lane[0-7]els_input_power_lane[0-7]laser_ag
PEMI Laser Source Advanced	health_value_laser_lane[0-7] health_value_tec_lane[0-7]laser_mpd_lane[0-7]power_consumptiontec_vo
PEMI Laser Source Essential	bias_current_monitor_lane[0-7] icc_monitoropt_power_monitor_lane[0-7]voltage_monitor_lane[0-7]
PEMI Module Status Properties	max_tec_power_high_alarm max_tec_power_high_warningmax_tec_power_low_alarmmax_tec_power_l
PEMI Pam4 Samples	pam4_level_transition_host_lane[0-7] pam4_level_transition_media_lane[0-7]
PEMI Pre FEC BER Samples	pre_fec_ber_avg_host pre_fec_ber_avg_mediapre_fec_ber_max_hostpre_fec_ber_max_mediapre
PEMI Pre FEC BER Properties	pre_fec_ber_avg_high_host_alarm pre_fec_ber_avg_high_host_warningpre_fec_ber_avg_high_media_alarmpr
PEMI Ferc Samples	ferc_avg_host ferc_avg_mediaferc_max_hostferc_max_mediaferc_min_hostferc_min_me
PHY Counters	link_down_events
PHY Statistics	conf_level_raw_ber eff_bereffective_ber_coefeffective_ber_magnitudephy_corrected_bitsphy_

Register / Page Name	Field List
PHY InfiniBand General Counters	dqs2llu_xmit_wait_arb_global general_receive_discard_external_containgeneral_transmit_discard_exterr
Histograms	eeber fc_zero_histhist[0-15]
Troubleshoot	group_opcode status_messagestatus_opcodeuser_feedback_datauser_feedback_index
Operation Info	cable_link_speed_cap cable_link_width_capcable_proto_cap_extcore_to_phy_link_proto_enabled
Link Down Info	cons_eff_norm_ber cons_raw_norm_bercons_symbol_norm_berdown_blamee2e_reason_opco
Link Up Info	fast_link_up_status lt_cnttime_logical_init_to_activetime_of_module_conf_done_downtime_of
PLR	plr_codes_loss plr_rcv_code_errplr_rcv_codesplr_rcv_uncorrectable_codeplr_sync_events
Port VL	PortRcvDataSLExt[0-15] PortRcvDataVLExt[0-15]PortRcvPktVLExt[0-15]PortVLXmitFlowCtlUpdate
Congestion Control	PortVLXmitTimeCongExt[0-15]
MLNX Counters Page0	rq_num_dup rq_num_laerq_num_leeoerq_num_llerq_num_lperq_num_lqpoerq_num_mc
MLNX Counters Page1	max_cnak_fifo_size maximum_dcrsminimum_dcrsrq_close_non_gb_gcrq_curr_gb_connectrq_
MLNX Counters Page255	dc_has_max_used_entries_in_line dc_hash_curr_used_linesnum_cqovfnum_eqovfrq_close_gb_gcrq_cwc_gh
Sharp PM Counters	ack_packet_sent aeth_syndrome_ack_packetethba_multi_packet_message_dropped_bytesht
General Info	build_id device_hw_revdevice_iddevice_technologyfw_date_dmyfw_hourfw_ini_ver

Register / Page Name	Field List
Device Temperature	dev_temperature
Port Counters	excessive_buffer_errors link_down_counterlink_error_recovery_counterlocal_link_integrity_errorspc
Port Counters Extended	ExcessiveBufferOverrunErrorsExtended LinkDownedCounterExtendedLinkErrorRecoveryCounterExtendedLocalLinl
Extended Speeds Counters	ErrorDetectionCounterLane_[0-11] FECCorrectableBlockCountrLane_[0-11]FECCorrectedSymbolCounterLane.
LLR Statistics	MaxRetransmissionRate
Port Rcv Error Details	PortBufferOverrunErrors PortDLIDMappingErrorsPortLocalPhysicalErrorsPortLoopingErrorsPortMalf
Port Xmit Discard Details	PortInactiveDiscards PortNeighborMTUDiscardsPortSwHOQLifetimeLimitDiscardsPortSwLifetim
Port RN Counters	pfrn_received_error pfrn_received_packetpfrn_start_packetpfrn_xmit_packetport_ar_trialspor
Port HBF Counters	rx_pkt_forwarding_ar rx_pkt_forwarding_ar_sg[0-2]rx_pkt_forwarding_hbfrx_pkt_forwarding_hk
Fast Recovery Counters	consecutive_normal_credit_watchdog consecutive_normal_eff_berconsecutive_normal_raw_berconsecutive_nori
Recovery Policy Type Counters	last_result_[0-7] last_time_spent_[0-7]rp_time_since_last_recoverytotal_success_[0-7]tota
Recovery Policy Type Config	draining_timeout_[0-7] link_down_timeout_[0-7]recovery_type_capability_[0-7]recovery_type_en_l
PPCNT Recovery Counters	host_logical_recovery_count host_logical_succesful_recovery_counthost_serdes_freq_recovery_countho
Port General Counters	contain_n_drain_rcv_discards contain_n_drain_xmit_discardsrx_icrc_errortx_parity_error

Register / Page Name	Field List
Port Hierarchy	bus devicefunctionm_asicm_asic_namem_board_typem_cagem_chassis_slot_i
Link Partner	link_partner_description link_partner_lidlink_partner_node_guidlink_partner_port_numlink_partner
Ports	fec_mode_active link_speed_activelink_speed_enabledlink_speed_supportedlink_width_acti
Calculated Info	NormalizedXW Normalized_CBWNormalized_XmitDatainfiniband_CBW
Mad Statistics	recv_time_port_counters recv_time_port_counters_extendedrecv_time_port_extended_speedsrecv_
Performance Histogram Info	cap_cell_size cap_hist_bin_sizecap_max_port_hist_idcap_max_sample_time
Performance Histogram Buffer Control	phb_control_bin_size_egress[0-15] phb_control_bin_size_ingress[0-15]phb_control_hist_min_value_egress[0-
Performance Histogram Buffer Data	phb_data_bin[0-9]_egress[0-15] phb_data_bin[0-9]_ingress[0-15]phb_data_histogram_type_egress[0-15]p
Performance Histogram Ports Control	php_bin_size_id[0-1] php_hist_min_value_id[0-1]php_histogram_type_id[0-1]php_mode_id[0-1]
Performance Histogram Ports Data	php_data_bin[0-9]_id[0-1] php_data_histogram_type_id[0-1]php_data_max_sampled_id[0-1]php_dat

Cable Info

Telemetry supports the following field list for cable information.

node_guid

port_guid

sys_image_guid

Port

aport

Lid

node_type

port_type

num_planes

Port_Name

Vendor

vendor_oui

PN

SN

Rev

smf_length

LengthCopperOrActive

Identifier

cable_identifier

Connector

Type

ib_compliance_code

length

length_by_pttl

length_by_reg

cable_type
SupportedSpeedDesc
Temperature
PowerClass
CDREnableTxRx
tx_cdr_enable
rx_cdr_enable
cable_tx_equalization
cable_rx_amp
OutputEmp
cable_rx_emphasis
cable_rx_post_emphasis
fw_version
mi_rx_power_type
HighTemperatureAlarm
LowTemperatureAlarm
HighTemperatureWarning
LowTemperatureWarning
InitializationFlagComplete
HighSupplyVoltageAlarm
LowSupplyVoltageAlarm
HighSupplyVoltageWarning
LowSupplyVoltageWarning

diag_supply_voltage
transmitter_technology
ActiveWavelengthControl
CooledTransmitterDevice
ActivePinDetector
TunableTransmitter
ExtendedSpecificationComplianceCodes
temperature_high_th
temperature_low_th
WarnTemperatureHighThresh
WarnTemperatureLowThresh
voltage_high_th
voltage_low_th
WarnVoltageHighThresh
WarnVoltageLowThresh
rx_power_high_th
rx_power_low_th
tx_power_high_th
tx_power_low_th
tx_bias_high_th
tx_bias_low_th
date_code
Lot

RXOutputDisable
TXAdaptiveEqualizationEnable
max_power
cable_vendor
ib_width
wavelength
ethernet_compliance_code
cable_breakout
tx_cdr_cap
rx_cdr_cap
memory_map_rev
wavelength_tolerance
module_st
rx_output_valid
rx_input_valid
active_set_host_compliance_code
active_set_media_compliance_code
tx_input_freq_sync
error_code
did_cap
cdr_vendor
max_fiber_length
dp_fw_fault

mod_fw_fault
tx_fault
rx_output_valid_change
rx_input_valid_change
temperature_alarm_and_warning
voltage_alarm_and_warning
tx_los
tx_cdr_lol
tx_ad_eq_fault
tx_power_hi_al
tx_power_lo_al
tx_power_hi_war
tx_power_lo_war
tx_bias_hi_al
tx_bias_lo_al
tx_bias_hi_war
tx_bias_lo_war
rx_los
rx_cdr_lol
rx_power_hi_al
rx_power_lo_al
rx_power_hi_war
cable_temperature

link_partner
HighRX[1-4]PowerAlarm
HighRX[1-4]PowerWarning
HighTX[1-4]BiasAlarm
HighTX[1-4]BiasWarning
HighTX[1-4]PowerAlarm
HighTX[1-4]PowerWarning
LengthOM[1-5]
LowRX[1-4]PowerAlarm
LowRX[1-4]PowerWarning
LowTX[1-4]BiasAlarm
LowTX[1-4]BiasWarning
LowTX[1-4]PowerAlarm
LowTX[1-4]PowerWarning
RX[1-4]CDRLOL
RX[1-4]LatchedLossIndicator
TX[1-4]AdaptiveEqualizationFaultIndicator
TX[1-4]AdaptiveEqualizationFreeze
TX[1-4]CDRLOL
TX[1-4]LatchedLossIndicator
cable_attenuation_2_5g
cable_attenuation_[5,7,12,25]g
dp_st_lane[0-3]

nbr_250

node_guid_plane_[0-3]

port_num_plane_[0-3]

rx_power_lane[0-7]

tx_bias_lane[0-7]

tx_power_lane[0-7]

Supported Docker Statistics

- mem_buffers – relatively temporary storage for raw disk blocks that should not become exceptionally large
- mem_cached – memory in the pagecache (diskcache) minus SwapCache—does not include SwapCached
- mem_free – sum of free lowmem and free highmem
- mem_swap_chache – memory that was once swapped out is swapped back in but is still kept in the swap file
- mem_total – total usable RAM
- mlnx:total_read_time – time spent on reading all counters
- clx_cpu_load
- clx_pid
- clx_res_mem
- clx_shr_mem
- clx_virt_mem

Type	Field
cable	timestamp node_guidport_guidLidActivePinDetectorActiveWavelengthControlCDREnableTxRx 4]PowerAlarmLowRX[1-4]PowerWarningLowSupplyVoltageAlarmLowSupplyVoltage

Type	Field
	4]LatchedLossIndicatorTemperatureTunableTransmitterTypeVendorWarnTemperat 3]error_codeethernet_compliance_codefw_versionib_compliance_codeib_widthler 3]rx_power_lo_alrx_power_low_thsmf_lengthtemperature_alarm_and_warningtem 3]tx_power_lo_altx_power_lo_wartx_power_low_thvendor_ouivoltage_alarm_and_v

Switch Info

The following table lists the supported register names by telemetry, along with their respective field lists.

Register Name	Field List
temp_fields	node_guid sensor_indexmtmp_sensor_nametemperaturemax_tem
power_fields	node_guid sensor_indexmvcr_sensor_namevoltagecurrentfeed_ma
total_power_managed_fields	node_guid total_power_managed
power_supplies_fields	node_guid psu_idxac_statusdc_statusalert_psupresent_psfan_ps
managed_power_supplies_fields	node_guid psu_idxpower_managed_supvoltage_managed_supcurr
fan_fields	node_guid sensor_indexfan_speedfan_speed_f[1-2]_managed
general_fields	node_guid serial_numberpart_numberrevisionproduct_namerando

PCI Info

The following field list of PCI info is supported by telemetry.

node_guid

pci_node

pcie_index

depth
fw_version
device_id
device_technology
mvcr_sensor_name
mtmp_sensor_name
switch_voltage
switch_current
switch_temperature
switch_serial_number
switch_part_number
capability_mask
link_speed_enabled
link_width_enabled
link_speed_active
link_width_active
num_of_vfs
num_of_pfs
lane_reversal
port_type
pwr_status
max_payload_size
max_read_request_size

pci_power
link_peer_max_speed
port_state
device_status
receiver_detect_result
life_time_counter
life_time_counter_low
life_time_counter_high
rx_errors
tx_errors
crc_error_dllp
crc_error_tlp
tx_overflow_buffer_pkt
tx_overflow_buffer_marked_pkt
outbound_stalled_reads
outbound_stalled_writes
outbound_stalled_reads_events
outbound_stalled_writes_events
effective_ber_coef
effective_ber_magnitude
effective_ber_pci
time_since_last_clear_high
time_since_last_clear_low

life_time_counter_high_lanes
life_time_counter_low_lanes
life_time_counter_high_timers
life_time_counter_low_timers
time_to_boot_image_start
time_to_link_image
calibration_time
time_to_first_perst
time_to_detect_state
time_to_crs_en
time_to_plastic_image_start
time_to_iron_image_start
perst_handler
dl_down
correctable_err_msg_sent
non_fatal_err_msg_sent
fatal_err_msg_sent
bdf0
config_cycle16to63usec
config_cycle2to7usec
config_cycle8to15usec
config_cycle[1,64]usec
error_counter_lane[0-15]

l0_to_recovery_eieos

l0_to_recovery_framing

l0_to_recovery_retrain

l0_to_recovery_ts

lane0_physical_position

time_to_l0

times_in_l[1,23]

NVLink Info

The table below lists the supported register names by telemetry, along with their corresponding field lists.

Register Name	Field List
nvl_class_port_info_fields	node_guid CapMskClassVersionBaseVersionRespTimeValueRedire
nvl_hbf_config_fields	node_guid port_numhash_typepacket_hash_bitmaskseedfields_e
nvl_reduction_port_info_fields	node_guid port_numport_direction_is_upexclude_include_self
nvl_reduction_config_mlid_fields	node_guid port_nummlid[0-7]
nvl_reduction_info_fields	node_guid reduction_FDB_Caphbf_group_capreduction_fdb_topn
nvl_cdrain_info_fields	node_guid port_guidport_numdi_ingress_port_statedi_egress_po
nvl_cdrain_port_state_fields	node_guid port_guidport_numdps_ingress_port_statedps_egress.
nvl_reduction_counters_fields	node_guid port_guidport_numblocklean_pipe_selectmlidincoming
nvl_reduction_rounding_fields	node_guid native_8_bitmixed_8_bitnative_16_bitmixed_16_bitdet

Documentation History

- [Document Revision History](#)
- [Release Notes History](#)

Document Revision History

Version	Date	Description
Rev 1.23.1	Nov 10, 2025	Updated: <ul style="list-style-type: none">• Changes and New Features in This Release• System Requirements• Bug Fixes in This Release• Ports Counters Info
Rev 1.22.2	Sep 5, 2025	UFM Telemetry v1.22.2 Documentation History
Rev 1.22.0	Aug 7, 2025	UFM Telemetry v1.22.0 Documentation History
Rev 1.21.0	May 5, 2025	UFM Telemetry v1.21.0 Documentation History
Rev 1.20.1	Feb 10, 2025	UFM Telemetry v1.20.1 Documentation History
Rev 1.19.15	Jan 9, 2025	UFM Telemetry v1.19.5 Documentation History
Rev 1.19.14	Dec 31, 2024	UFM Telemetry v1.19.4 Documentation History
Rev 1.19.1	Dec 6, 2024	UFM Telemetry v1.19.1 Documentation History
Rev 1.19.0	Nov 7, 2024	UFM Telemetry v1.19.0 Documentation History
Rev 1.18.2	Aug 12, 2024	UFM Telemetry v1.18.2 Documentation History
Rev 1.17.0	May 5, 2024	UFM Telemetry v1.17.0 Documentation History
Rev 1.16.5	Feb 2024	UFM Telemetry v1.16.5 Documentation History
Rev 1.15.6	Dec 2023	UFM Telemetry v1.15.6 Documentation History
Rev 1.15	Nov 2023	UFM Telemetry v1.15.0 Documentation History
Rev 1.14.5	Aug 2023	UFM Telemetry v1.14.5 Documentation History

Version	Date	Description
Rev 1.14	Aug 2023	UFM Telemetry v1.14.0 Documentation History

Release Notes History

- [Bug Fixes History](#)

Bug Fixes History

Revision	Date	Description
Rev 1.22.0	Aug 7, 2025	Bug Fixes History in UFM Telemetry v1.22.0
Rev 1.21.0	May 4, 2025	Bug Fixes History in UFM Telemetry v1.21.0
Rev 1.20.1	Feb 10, 2025	Bug Fixes History in UFM Telemetry v1.20.1
Rev 1.19.15	Jan 9, 2025	Bug Fixes History in UFM Telemetry v1.19.15
Rev 1.19.14	Dec 31, 2024	Bug Fixes History in UFM Telemetry v1.19.4
Rev 1.19.1	Dec 6, 2024	Bug Fixes History in UFM Telemetry v1.19.1
Rev 1.19.0	Nov 7, 2024	Bug Fixes History in UFM Telemetry v1.19.0
Rev 1.18.2	Aug 12, 2024	Bug Fixes History in UFM Telemetry v1.18.2
Rev 1.17.0	May 5, 2024	Bug Fixes History in UFM Telemetry v1.17.0
Rev 1.15.6	Dec 2023	Bug Fixes History in UFM Telemetry v1.15.6
Rev 1.15	Nov 2023	Bug Fixes History in UFM Telemetry v1.15.0
Rev 1.14.5	Aug 2023	Bug Fixes History in UFM Telemetry v1.14.5

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF

ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2025, NVIDIA. PDF Generated on 11/17/2025