NVIDIA UFM Telemetry Documentation v1.15.6

Exported on Dec/11/2023 03:20 PM
https://docs.nvidia.com/networking/x/uyqgC

Table of Contents

⚠ You can download a PDF version here.

About This Document

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.

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: https://www.nvidia.com/en-us/support/enterprise

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.

Changes and New Features in This Release

New Features in v1.15

- Updated show switch routing decision to AR/HBF
- Exposed SHARP telemetry
- Added support for MSPS updated register

Changes in v1.15

Added `m_key` option to the bringup `collection_start` command. This option is relevant for clusters with non-default m_key value.

System Requirements

Platform	Type and Version
OS and Kernel	• RedHat 7.X • RedHat 8.X • RedHat 9.X • Ubuntu18 • Ubuntu20 • Ubuntu22
CPU	x86_64

Platform	Type and Version
OFED	MLNX_OFED 5.X

Bug Fixes in This Release

Ref. #	Description
3667820	NormalizedXmitData 100% - miscalculated
	Keyword: NormalizedXmitData 100%, miscalculated field
	Discovered in release: 1.14

Known Issues in This Release

Ref. #	Description
3689874	**Description**: Faulty prometheus format
	Workaround: Remove the following field from xcset: link_partner_node_guid, Time_since_last_clear_[Min], Speed_[Gb/s] (on default part of “low_freq_debug” and “amber”)
	Keywords: Prometheus, Wrong format

Bug Fixes History

Ref. #	Description
3597364	Description: Multi-rate usage causes a crash
	Keywords: Coredump, Multi-rate
	Discovered in release: 1.14
3648187	Description: Hundreds of 'could not retrieve data for' warning messages
	Keywords: Warn, Warning
	Discovered in release: 1.14
3590777	**Description**: After upgrading UFM new telemetry data is not being collected and presented in UI Telemetry tab.
	Keywords: Telemetry, Coredump
	Discovered in release: 1.14
3442081	**Description**: The gen_metadata script fails and cannot generate metadata file.
	Keywords: Metadata, Labels
	Discovered in release: 1.12
3331553	**Description:** [installation]: no check for supervisord as a prerequisite.
	Keywords: run_bringup, supervisord
	Discovered in release: 1.12

Ref. #	Description
3546901	**Description:** Invalid Cable Power data retrieved
	Keywords: Cable power, Power infinity
	Discovered in release: 1.13
3481178	Description: Fixed UFM-Telemetry data refresh delay.
	Keywords: Data Delay; Refresh;
	Discovered in release: 1.13.5
3477852	Description: Calculated values are set to (-1) when unavailable.
	Keywords: Unavailable Data;
	Discovered in release: 1.13.5
3481011	Description: Fixed cable voltage values presentation in microvolts (uV) instead of millivolts (mV).
	Keywords: Cable; Voltage; Unit of measurement;
	Discovered in release: 1.13.5
3438270	Description: Added support to starting UFM telemetry using the bringup tool with multi_port_sm configured.
	Keywords: Multiple HCAs; HCA list;
	Discovered in release: 1.13.5
3488824	Description: Fixed MADs statistics clearance between iterations.
	Keywords: MAD Statistics;
	Discovered in release: 1.13.5
3477561	Description: Fixed faulty Telemetry results in segmentation.
	Keywords: segfault;
	Discovered in release: 1.13.5
3461058	Description: UFM Telemetry log rotation applications are not used

Ref. #	Description
	Keywords: Log, rotation applications
	Discovered in release: 1.13.0
3397908	Description: UFM Telemetry fails to start due to supervisord
	Keywords: supervisord, start failure
	Discovered in release: 1.12.0
3440964	Description: _MEI folders created in tmp are not removed
	Keywords: _MEI, temp folders
	Discovered in release: 1.12.0
3214566	Description: Dynamic Fabric unexpected header warning
	Keywords: Log warning, Dynamic Fabric
	Discovered in release: 1.10
3292572	Description: [systemd service]: executable path is not absolute
	Keywords: systemd
	Discovered in release: 1.11
3305919	Description: Timestamp error in CSV serialize of fset
	Keywords: timestamp
	Discovered in release: 1.11
3306476	Description: HTTP endpoint: fset "[CableInfo]Temperature" is not rendered
	Keywords: HTTP endpoint, Cable temperature

Ref. #	Description
	Discovered in release: 1.11
3327188	Description: Missing amBER configuration for fluentbit
	Keywords: amber, Container configuration
	Discovered in release: 1.11
3327193	Descrition: CSV output contains redundant comma at EOL
	Keywords: csv
	Discovered in release: 1.11
3332112	Description: Failed to execute script 'gen_metadata'
	Keyword: get_metadata
	Discovered in release:1.11
3229888	Description: Error on launch_ibdiagnet.log
	Keywords: Log Error
	Discovered in release: 1.10
3225679	Description: Adding eff_ber counter to low_freq.cset file
	Keywords: cset, eff_ber
	Discovered in release: 1.10
N/A	Description: The Temperature fset does not contain labels
	Keywords: Temperature, labels

Ref. #	Description
	Discovered in release: 1.9
N/A	Description: Cannot rename fields in fset
	Keywords: fset, events
	Discovered in release: 1.9
N/A	Description: The node_guid field in fsets is missing leading zeros
	Keywords: fset, events
	Discovered in release: 1.9
3051843	Description: Can't start telemetry without setting arg_12= to empty
	Keywords: startup failure
	Discovered in release: 1.9
3076045	Description: gen_metadata doesn't use a rules file when provided
	Keywords: metadata, labels
	Discovered in release: 1.9
2943459	Description: Custom labels lost after UFM restart
	Keywords: UFM restart, labels
	Discovered in release: 1.8
2921452	Description: raw_ber from UFM Telemetry doesn't match output of mlxlink -e -c -m

Ref. #	Description
	Keywords: BER
	Discovered in release: 1.8
2923525	Description: Enabling BER collection on an NDR fabric causes segmentation fault of ibdiagnet
	Keywords: BER, ibdiagnet, segmentation fault
	Discovered in release: 1.8
3037715	Description: ConnectX-7 infiniband_CBW, Normalized_CBW are always zero
	Keywords: InfiniBand
	Discovered in release: 1.8
3018638	Description: Normalized TX data returned 0% for ConnectX-7
	Keywords: TX data
	Discovered in release: 1.8

Ref. #	Description

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 three 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:
    a. Python3
    b. Python3-venv
    c. 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:
                        options                          defaults
                        -------                          --------
      collection_start  time|duration=n [s|m|h|d]        24h
                        sample_rate=n [s|m|h|d]          60 seconds
                        guids=[guid_list|guid_file]      None
                        hca=hca_name                     mlx5_0
                        cable|cable_info=[yes|no|once]   yes
                        reset_counters=t                 false
                        mads_retries=n                   2
                        mads_timeout=n (msec)            500
                        force_hca=t                      f
```

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=1/00:00,3/00:00,5/00:00"
```

> ⚠ 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.
   a. Make sure the UFM Telemetry container is up.

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

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

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

   c. 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.

⚠ 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:

⚠ 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
```

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

⚠ 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:
   a. Python3
   b. Python3-venv
   c. 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_CLX_FILE_WRITE_ENABLED=1"
```

⚠ 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:
    a. Python3
    b. Python3-venv
    c. 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_CLX_FILE_WRITE_ENABLED=1" --gen_systemd_service
```

⚠ 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:

⚠ 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
```

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

⚠ 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 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,length,type,supportedspeed,temperature,powercla
ss,
nominalbitrate,cdrenabletxrx,inputeq,outputamp,outputemp,fw_version,attenuation_2.5_5_7_12,rx_power_type,
rx_power.1.mw,rx_power.1.dbm,rx_power.2.mw,rx_power.2.dbm,rx_power.3.mw,rx_power.3.dbm,rx_power.4.mw,
rx_power.4.dbm,tx_bias.1,tx_bias.2,tx_bias.3,tx_bias.4,tx_power.1.mw,tx_power.1.dbm,tx_power.2.mw,tx_power.2.dbm,t
x_power.3.mw,tx_power.3.dbm,tx_power.4.mw,tx_power.4.dbm,cdr_tx_rx_loss_indicator,adaptive_equalization_fault,tx_r
x_lol_indicator,temperature_alarm_and_warning,voltage_alarm_and_warning,rx_power_alarm_and_warning,tx_bias_alarm_a
nd_warning,tx_power_alarm_and_warning,diag_supply_voltage,transmitter_technology,eth_com_codes_ext,datacode,lot,tx
_adaptive_equalization_freeze,rx_output_disable,tx_adaptive_equalization_enable,
2020-07-26T15:00:12.742710,cable_info,1595764812742710,1,117,0x248a0703008b20ec,ufm-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,-999.999023438,*,*,*,*,0.0,-999.999023438,0.0,-
999.999023438,0.0,-999.999023438,0.0,-999.999023438,0,0,0,0,0,0,0,0,160,0,14-11-27,8224,0,0x0,0x0,
2020-07-26T15:00:12.742710,cable_info,1595764812742710,1,104,0xe41d2d0300109610,msib-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,-999.999023438,*,*,*,*,0.0,-999.999023438,0.0,-
999.999023438,0.0,-999.999023438,0.0,-999.999023438,0,0,0,0,0,0,0,0,160,0,14-11-27,8224,0,0x0,0x0,
2020-07-26T15:00:12.742710,cable_info,1595764812742710,3,104,0xe41d2d0300109610,msib-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,-999.999023438,*,*,*,*,0.0,-999.999023438,0.0,-
999.999023438,0.0,-999.999023438,0.0,-999.999023438,0,0,0,0,0,0,0,0,160,0,14-03-25,8224,0,0x0,0x0,
2020-07-26T15:00:12.742710,cable_info,1595764812742710,1,187,0xe41d2d03005d2250,ip-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,-999.999023438,*,*,*,*,0.0,-999.999023438,0.0,-
999.999023438,0.0,-999.999023438,0.0,-999.999023438,0,0,0,0,0,0,0,0,160,0,14-03-25,8224,0,0x0,0x0,
```

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:

```
CollectX: help port_counters
```

```
Usage:
            port_counters [TAGS] [TIME] [out=]

            [TAGS] is a list of countersets/name fragments.
            [TIME] can be specified as:
                                    date=
                                    past=n[hours|days]
                                    from= to=
            [out=] to specify output file (optional)

Description:
            Dump port_counters matching a given fragment/counterset for a
            given date or range of dates.

Example:
            port_counters error past=10m
            port_counters error date=jul16 out=error_dump.csv
```

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

```
CollectX: port_counters past=300s

idx,time,ts,node,port_num,PortMultiCastXmitPktsExtended,PortUniCastXmitPktsExtended,PortXmitConstraintErrorsExtend
ed,
PortXmitDataExtended,PortXmitDiscardsExtended,PortXmitPktsExtended,PortXmitWaitExtended,port_xmit_constraint_error
s,
port_xmit_data,port_xmit_discard,port_xmit_pkts,port_xmit_wait,
0,2020-07-27T09:57:02,1595833022873349,0xb8599f0300355d6e,1,0,0,0,0,0,0,0,0,0,0,0,0,
1,2020-07-27T09:57:02,1595833022873363,0xb8599f0300355d6e,2,0,0,0,0,0,0,26700642,0,0,0,0,26700642,
2,2020-07-27T09:57:02,1595833022873374,0xb8599f0300355d6e,3,0,9881980474,0,4540230161607,0,9881980474,0,0,0,0,0,0,
3,2020-07-27T09:57:02,1595833022873396,0xb8599f0300355d6e,4,0,0,0,1339496959094,0,0,21581722,0,0,0,0,21581722,
4,2020-07-27T09:57:02,1595833022873408,0xb8599f0300355d6e,9,24766362454,0,0,0,0,0,0,0,0,0,0,0,
5,2020-07-27T09:57:02,1595833022873419,0xb8599f0300355d6e,10,0,54725808986,0,8222892412792,0,29959446532,33957262,
0,0,0,0,33957262,
```

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

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

```
CollectX: switch_temperature past=10m out=switch_temperature.csv
time,source,timestamp,node_guid,sensor_index,mtmp_sensor_name,temperature,max_temperature,
12T17:05:16.332772,0xe41d2d030003e450,1649783116332772,0xe41d2d030003e450,0,,47,51,
2022-04-12T17:05:16.332772,0xe41d2d030003e450,1649783116332772,0xe41d2d030003e450,1,,30,33,
2022-04-12T17:05:16.332772,0xe41d2d030003e450,1649783116332772,0xe41d2d030003e450,2,,33,37,
2022-04-12T17:05:16.332772,0xec0d9a0300b41a50,1649783116332772,0xec0d9a0300b41a50,0,,58,66,
2022-04-12T17:05:16.332772,0xec0d9a0300b41a50,1649783116332772,0xec0d9a0300b41a50,1,,27,31,
2022-04-12T17:05:16.332772,0xec0d9a0300b41a50,1649783116332772,0xec0d9a0300b41a50,2,,33,37,
...
```

Switch Fans

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

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

```
CollectX: help switch_fans
Usage:
            switch_fans  [TIME] [out=]
            [TIME] is one of the following:
                                              last
                                              date=
                                              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,
2020-10-04T17:36:05.287397,0xe41d2d0300169e40,1601822165287397,0xe41d2d0300169e40,1,10288,
2020-10-04T17:36:05.287402,0xe41d2d0300169e40,1601822165287402,0xe41d2d0300169e40,2,8823,
2020-10-04T17:36:05.287403,0xe41d2d0300169e40,1601822165287403,0xe41d2d0300169e40,3,10608,
2020-10-04T17:36:05.287404,0xe41d2d0300169e40,1601822165287404,0xe41d2d0300169e40,4,9118,
…
```

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,product_name,random_fdb_cap,linear_fdb_cap,line
ar_fdb_top,mcast_fdb_cap,optimized_slvl_mapping,port_state_change,life_time_value,def_mcast_not_pri_port,def_mcast
_pri_port,def_port,part_enf_cap,lids_per_port,mcast_fdb_top,enp0,filter_raw_outb_cap,filter_raw_inb_cap,outb_enf_c
ap,inb_enf_cap,
2020-10-25T11:41:05.183039,0xe41d2d0300169e40,1603618865183039,0xe41d2d0300169e40,MT1510X10802,MSB7700-
EB2F,A6,Scorpion IB EDR,0,49152,7936,16383,1,1,19,255,255,0,32,0,49183,1,1,1,1,1,
2020-10-25T11:42:05.559284,0xe41d2d0300169e40,1603618925559284,0xe41d2d0300169e40,MT1510X10802,MSB7700-
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_configs
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.

Ibdiagnet 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 filed/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 fiedls are user-defined additional fiedls 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 apper 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 sting "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 fiedls are user-defined additional fiedls 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 apper 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 sting "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_bit_configs
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.

> ⚠ 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_counterm 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	Description
ibdiagnet	ibdiagnet_enabled	bool	true	Enable/disable run ibdiagnet process
	data_dir	String	/data	Directory in which UFM Telemetry data is placed
	ibdiag_output_dir	String	/tmp/ibd	Directory in which ibdiagnet places files
	sample_rate	Int	-	Frequency of collecting ports counters data
	hca	String	mlx5_2	Card to use. Can provide a comma-separated list of cards for local high availability
	force_hca	bool	false	Skip hca state check
	app_name	String	/opt/collectx/bin/ibdiagnet	Allow user to specify full path of the ibdiagnet application if necessary
	topology_mode	String	discover	Topology policy
	topology_discovery_factor	Int	0	Every "n" iterations, do discovery, otherwise, use result from last run if 0 or 1
Retention	retention_enabled	bool	true	Enable/disable retention service
	retention_interval	time	1d	Interval to wait before running the retention process
	retention_age	time	100d	Period to reserve the collected data
compression	compression_enable	bool	true	Enable/disable compression service
	compression_interval	time	6h	Interval to wait before running the compression service
	compression_age	time	12h	Period to reserve the compressed data

Section	Key	Type	Default Value	Description
cable_info	cable_info_schedule	CSV	-	weekday/hr:min,hr:hm Time to collect cable info data

Enable BER Collection

To enable the BER collection, make sure the following lines appear and are not commented out. Specifically, the `--enabled_regs dd_ppcnt_plsc` needs to be added.

```
lookup_BER_counters=--get_phy_info --enabled_regs dd_ppcnt_plsc
param_4=BER_counters
```

Verify that the following flag is commented out or set to 0 (default is 1):

```
plugin_env_CLX_EXPORT_API_SKIP_PHY_STAT
```

Enable Temperature Collection

Comment out the following line to make sure temperature sensing will not be skipped:

```
# arg_13=--skip temp_sensing
```

Enable Grade Collection

To enable the BER collection, make sure the following lines appear and are not commented out. Specifically, the `--enabled_regs dd_ppcnt_plsc` needs to be added.

```
lookup_Grade_counters=--get_phy_info --enabled_regs slrg
param_6=Grade_counters
```

Verify that the following flag is commented out or set to 0 (default is 1):

```
plugin_env_CLX_EXPORT_API_SKIP_SLRG
```

Enable PPCC

To enable PPCC, ensure that the following line is added and not commented:

```
arg_x=--congestion_counters # x should be replaced with the next available index!
```

Verify that the following flag is set to 0:

```
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`

Enable XMIT_WAIT per vl

To enableXMIT_WAIT per vl, ensure that the following line is added and not commented:

```
arg_x=--per_slvl_cntrs #  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_SKIP_PORT_VL=1
```

The following counters are created:
`PortXmitWaitVLExt[0-15]`

Enable MLNX_COUNTERS

To enable MLNX_COUNTERS (page0, 1, 255), ensure that the following line is added and not commented:

```
arg_x=--sc #  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_SKIP_MLNX_COUNTER=0
plugin_env_CLX_EXPORT_API_SKIP_MLNX_COUNTERS_PAGE1=0
plugin_env_CLX_EXPORT_API_SKIP_MLNX_COUNTERS_PAGE255=0
```

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
```

SHARP HW Counters

To enable Sharp HW (PM) counters, ensure the following line is added and not commented:

```
arg_x=--sharp -sharp_opt dsc # x should be replaced with next available index!
```

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

```
plugin_env_CLX_EXPORT_API_SKIP_SHARP_PM_COUNTERS=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 enables 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/management/key_value
```

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=11.222.33.44,11.333.444.55
  ```

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

  ```
  plugin_env_CLX_EXPORT_API_MANAGED_SWITCH_INTERVAL=600
  ```

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
```

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 `/xcet` can also be used to specify which filter file is meant.

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

⚠ 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.

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

Priority	Prefix	Link Examples	Description

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 filer file should be applied

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/gen_metadata --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/cset
plugin_env_PROMETHEUS_FSET_DIR=/telemetry.config/prometheus_configs/fset
plugin_env_PROMETHEUS_XCSET_DIR=/telemetry.config/prometheus_configs/xcset
```

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|SwitchIB Mellanox Technologies|device_type|switch|
fabric|compute|hostname||node_desc||level|leaf|peer_level|server
ec0d9a0300b41a50_37|port_id|ec0d9a0300b41a50_37|device_name|SwitchIB Mellanox Technologies|device_type|switch|
fabric|compute|hostname||node_desc||level|leaf|peer_level|
ec0d9a0300b41a58_1|port_id|ec0d9a0300b41a58_1|device_name||device_type|switch|fabric|compute|hostname|aggregation|
node_desc|aggregation node|level||peer_level|leaf
98039b0300640b92_1|port_id|98039b0300640b92_1|device_name||device_type|host|fabric|compute|hostname|agx-1|
node_desc|agx-1 mlx5_0|level|server|peer_level|leaf
98039b0300640c22_1|port_id|98039b0300640c22_1|device_name||device_type|host|fabric|compute|hostname|agx-2|
node_desc|agx-2 mlx5_0|level|server|peer_level|leaf
0002c90300f172a0_2|port_id|0002c90300f172a0_2|device_name||device_type|host|fabric|compute|hostname|agx-3|
node_desc|agx-3 mlx4_0|level|server|peer_level|leaf
98039b0300640b9a_1|port_id|98039b0300640b9a_1|device_name||device_type|host|fabric|compute|hostname|agx-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",hostname="jazz32",level="server",node_desc="j
azz32 mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"}  0 1629194120043
port_rcv_switch_relay_errors{device_name="",device_type="host",fabric="compute",hostname="jazz32",level="server",n
ode_desc="jazz32 mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"}  0 1629194120043
port_rcv_constraint_errors{device_name="",device_type="host",fabric="compute",hostname="jazz32",level="server",nod
e_desc="jazz32 mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"}  0 1629194120043
port_xmit_constraint_errors{device_name="",device_type="host",fabric="compute",hostname="jazz32",level="server",no
de_desc="jazz32 mlx5_2",peer_level="leaf",port_id="ec0d9a0300c04a54_1"}  0 1629194120043
```

Appendixes

Appendix – Supported Counters

Supported InfiniBand Counters

- Counter
- ExcessiveBufferOverrunErrorsExtended
- GradeID
- Lane0Grade
- Lane1Grade
- Lane2Grade
- Lane3Grade
- LinkDownedCounterExtended
- LinkErrorRecoveryCounterExtended
- LocalLinkIntegrityErrorsExtended
- MaxRetransmissionRate
- PortBufferOverrunErrors
- PortDLIDMappingErrors
- PortFECCorrectableBlockCounter
- PortFECCorrectedSymbolCounter
- PortFECUncorrectableBlockCounter
- PortInactiveDiscards
- PortLocalPhysicalErrors
- PortLoopingErrors
- PortMalformedPacketErrors
- PortMultiCastRcvPktsExtended
- PortMultiCastXmitPktsExtended

- PortNeighborMTUDiscards
- PortRcvConstraintErrorsExtended
- PortRcvDataExtended
- PortRcvErrorsExtended
- PortRcvPktsExtended
- PortRcvRemotePhysicalErrorsExtended
- PortRcvSwitchRelayErrorsExtended
- PortSwHOQLifetimeLimitDiscards
- PortSwLifetimeLimitDiscards
- PortUniCastRcvPktsExtended
- PortUniCastXmitPktsExtended
- PortVLMappingErrors
- PortXmitConstraintErrorsExtended
- PortXmitDataExtended
- PortXmitDiscardsExtended
- PortXmitPktsExtended
- PortXmitWaitExtended
- QP1DroppedExtended
- RetransmissionPerSec
- SymbolErrorCounterExtended
- SyncHeaderErrorCounter
- UnknownBlockCounter
- VL15DroppedExtended
- ber_threshold
- eff_ber
- effective_ber_coef
- effective_ber_magnitude
- excessive_buffer_errors
- link_down_counter
- link_error_recovery_counter
- load_avg
- local_link_integrity_errors
- node_guid
- phy_corrected_bits

- phy_raw_errors_lane0
- phy_raw_errors_lane1
- phy_raw_errors_lane2
- phy_raw_errors_lane3
- phy_received_bits
- phy_symbol_errors
- port_guid
- port_num
- port_rcv_constraint_errors
- port_rcv_data
- port_rcv_errors
- port_rcv_pkts
- port_rcv_remote_physical_errors
- port_rcv_switch_relay_errors
- port_xmit_constraint_errors
- port_xmit_data
- port_xmit_discard
- port_xmit_pkts
- port_xmit_wait
- raw_ber
- raw_ber_coef
- raw_ber_magnitude
- symbol_error_counter
- threshold_type
- time_since_last_clear
- vl15_dropped

Supported Per-lane Counters

- ErrorDetectionCounterLane.<1-12>
- FECCorrectableBlockCountrLane.<1-12>
- FECCorrectedSymbolCounterLane.<1-12>
- FECUncorrectableBlockCounterLane.<1-12>

Appendix – Cable Information

Type	Field
cable	timestamp node_guid port_guid Lid ActivePinDetector ActiveWavelengthControl CDREnableTxRx Connector CooledTransmitterDevice ExtendedSpecificationComplianceCodes HighRX[1-4]PowerAlarm HighRX[1-4]PowerWarning HighSupplyVoltageAlarm HighSupplyVoltageWarning HighTX[1-4]BiasAlarm HighTX[1-4]BiasWarning HighTX[1-4]PowerAlarm HighTX[1-4]PowerWarning HighTemperatureAlarm HighTemperatureWarning Identifier InitializationFlagComplete LengthCopperOrActive LengthOM[1-5] Lot LowRX[1-4]PowerAlarm LowRX[1-4]PowerWarning LowSupplyVoltageAlarm LowSupplyVoltageWarning LowTX[1-4]BiasAlarm LowTX[1-4]BiasWarning LowTX[1-4]PowerAlarm LowTX[1-4]PowerWarning LowTemperatureAlarm LowTemperatureWarning OutputEmp

Type	Field
	PN Port Port_Name PowerClass RXOutputDisable RX[1-4]CDRLOL RX[1-4]LatchedLossIndicator Rev SN SupportedSpeedDesc TXAdaptiveEqualizationEnable TX[1-4]AdaptiveEqualizationFaultIndicator TX[1-4]AdaptiveEqualizationFreeze TX[1-4]CDRLOL TX[1-4]LatchedLossIndicator Temperature TunableTransmitter Type Vendor WarnTemperatureHighThresh WarnTemperatureLowThresh WarnVoltageHighThresh WarnVoltageLowThresh active_set_host_compliance_code active_set_media_compliance_code cable_attenuation_2_5g cable_attenuation_[5,7,12,25]g cable_breakout cable_identifier cable_rx_amp cable_rx_emphasis cable_rx_post_emphasis cable_temperature cable_tx_equalization cable_type cable_vendor date_code diag_supply_voltage did_cap dp_fw_fault dp_st_lane[0-3]

Type	Field
	error_code ethernet_compliance_code fw_version ib_compliance_code ib_width length link_partner max_power memory_map_rev mi_rx_power_type mod_fw_fault module_st nbr_250 rx_cdr_cap rx_cdr_enable rx_cdr_lol rx_input_valid rx_input_valid_change rx_los rx_output_valid rx_output_valid_change rx_power_hi_al rx_power_hi_war rx_power_high_th rx_power_lane[0-3] rx_power_lo_al rx_power_low_th smf_length temperature_alarm_and_warning temperature_high_th temperature_low_th transmitter_technology tx_ad_eq_fault tx_bias_hi_al tx_bias_hi_war tx_bias_high_th tx_bias_lane[0-3] tx_bias_lo_al tx_bias_lo_war tx_bias_low_th tx_cdr_cap

Type	Field
	tx_cdr_enable tx_cdr_lol tx_fault tx_input_freq_sync tx_los tx_power_hi_al tx_power_hi_war tx_power_high_th tx_power_lane[0-3] tx_power_lo_al tx_power_lo_war tx_power_low_th vendor_oui voltage_alarm_and_warning voltage_high_th voltage_low_th wavelength wavelength_tolerance

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

Document Revision History

Revision and Date	Description
Rev 1.15.6 - December 2023	Updated: • Bug Fixes in This Release • Known Issues in This Release • Software Management - Updated package paths and reorganized the chapter Added: • Bare Metal - Bringup Mode
Rev 1.15 - November 2023	Updated: • Changes and New Features in This Release • Bug Fixes in This Release • Settings and Configuration - Added `force_hca` key Added: • Switch Power Sensors Data • Switch Power Supplies Data • SHARP HW Counters
Rev 1.14.5 - August 2023	Updated Bug Fixes in This Release
Rev 1.14 - August 2023	Updated: • Changes and New Features in This Release • Bug Fixes in This Release • Extended Counter Set Filtering
Rev 1.13.7 - June 2023	Updated: • Changes and New Features in This Release • Bug Fixes in This Release

Revision and Date	Description
Rev 1.13.5 - May 2023	Updated: • Bug Fixes in This Release
Rev 1.13 - May 2023	Updated: • Changes and New Features in This Release • Bug Fixes in This Release • Data Filtering - Added `/xcset` filter file • Configuring Data Polling Endpoint • URL Prefixes Priority Added: • Managed Switch Data Collection • Extended Counter Set Filtering
Rev 1.12 - February 2023	Updated: • Changes and New Features in This Release • Bug Fixes in This Release • Updated Docker Container in Software Management • Docker Container Mode - High Availability • Cset Fset Filtering • Configuring Data Polling Endpoint • Appendix – Cable Information Added: • Deploying UFM Telemetry Bare Metal - High Availability • Enable MLNX_COUNTERS
Rev 1.11 - November 2022	Updated: • Changes and New Features in This Release • System Requirements • Bug Fixes in This Release • Upgrading UFM Telemetry Software to list instructions on all modes of operation • Software Management

Revision and Date	Description
	• Settings and Configuration Added: • Bug Fixes History • Enable PPCC • Enable XMIT_WAIT per vl • Log File Rotation • Docker Container Mode - High Availability
Rev 1.10 – July 2022	Updated: • Software Management
Rev 1.9 – May 2022	Added: • Section Deploying UFM Telemetry – Bringup Mode • Section Bringup – amBer Format Updated: • NVIDIA UFM TELEMETRY USER MANUAL • Software Management • Data Collection • Fluent Bit Export • Settings and Configuration • Prometheus Endpoint Support
Rev 1.8 – November 30, 2021	Added: • Bare Metal Mode • Container Mode • Deploying UFM Telemetry Bare Metal
Rev 1.7 – August 30, 2021	Added page Prometheus Endpoint Support
Rev 1.7 – March 29, 2021	Updated step 6 in section Deploying UFM Telemetry due to new licensing mechanism
Rev 1.7 – February 08, 2021	No changes

Revision and Date	Description
Rev 1.6 – January 27, 2021	Added: • Upgrading UFM Telemetry Software • Fluent Bit Export • UFM Telemetry Configuration Script • Switch Temperature • Switch Fans • Switch General • Appendixes
Rev 1.0 – August 27, 2020	First release

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. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any 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 NONINFRINGEMENT, 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.