NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP) Rev 3.15.0

Table of Contents

1 Overview 6

2 Document Revision History 7

3 Release Notes 8

3.1 General Information 8

3.1.1 Packages Provided with SHARP 8

3.1.2 Prerequisites 8

3.1.3 NVIDIA Hardware Capabilities and Limitations 9

3.1.4 Supported Operating Systems and Platforms 9

3.2 Changes and New Features 9

3.2.1 Changes and New Features 9

3.2.2 Parameter Changes 9

3.3 Bug Fixes in this Version 9

3.4 Known Issues 10

4 Introduction 13

5 NVIDIA SHARP Installation 14

5.1 Enabling SHARP in UFM 14

5.2 Installing the SHARP Client (libsharp) via HPC-X 14

5.3 Testing the Setup 14

5.3.1 Running Tests from the UFM Machine 15

5.3.2 NVIDIA SHARP Hello 15

6 Using SHARP 17

6.1 NVIDIA SHARP Collective Library 17

6.1.1 NVIDIA SHARP Library Flags 17

6.1.1.1 NVIDIA SHARP Configuration Flags 17

6.1.1.2 NVIDIA SHARP Resource Tuning for Low Latency Operations 18

6.1.1.3 NVIDIA SHARP Streaming Aggregation 18

6.1.1.4 SHARP Miscellaneous Tuning 19

6.2 Using NVIDIA SHARP with Open MPI 20

6.2.1 HCOLL Library Flags 20

6.2.1.1 Example of Allreduce with Default Settings with SHARP Enable 21

6.3 Using NVIDIA SHARP with NVIDIA NCCL 21

6.3.1 Requirements 21

6.3.2 Control Flags 22

6.3.3 Cluster Topology for Using NVIDIA SHARP SAT with NCCL 22
6.3.4 NCCL Benchmark Example 22
6.4 Using NVIDIA SHARP with UCC 23
6.4.1 UCC Library Flags 23
6.4.1.1 Example of Allreduce with Default Settings with SHARP Enable 24
7 SHARP in Public Cloud 25
7.1 Secret AM Key 25
7.2 SHARP PKeys Support 25
7.2.1 SHARP Resource Limits 26
7.3 Limitation 26
8 SHARP Monitoring 27
8.1 sharp_am Log and Dump Files 27
8.1.1 Activity Log Verbosity Level 27
8.1.2 Log Levels 27
8.1.3 Log Categories Config File 28
8.2 sharp_am Telemetry 28
8.3 NVIDIA SHARP Traps 30
8.4 Aggregation Trees Diagnostics 32
9 Appendices 34
9.1 Appendix A: Using SHARP with Quasi Fat Trees and DFP Topologies 34
9.2 Appendix B: SHARP_am Network Interfaces 35
9.2.1 Network Interfaces Configuration 35
9.2.2 Switching from UCX to Sockets 36
9.2.3 Management Host Network Interfaces High Availability 36
9.2.3.1 HA Configuration 37
9.3 Appendix C: HCA-to-TOR Benchmark Tool 37
9.3.1 Running the Benchmark Tool 37
9.3.1.1 Example Test Run Using the Host CPU 37
9.3.1.2 Example Test Run Using the Host GPU 38
9.3.1.3 Example Test Run Sending Data Directly from the HCA: 38
9.3.2 Operating in a PKEY-Based System 38
9.3.3 Limitations and Recommendations 38
9.4 Appendix D: Stochastic Rounding Support 38
9.5 Appendix E: NVIDIA SHARP Cleanup 39
9.5.1 Running the Cleanup in UFM 39

9.6 Appendix F: Running SHARP_am from HPC-X 39
9.7 Appendix G: Disabling SHARP on Specific Network Devices in OpenSM 40
10 Deployment Guide Revision History 41
11 Release Notes Revision History 42
11.1 Release Notes Change History 42
11.1.1 Changes and New Features History 42
11.1.2 Parameters Change History 47
11.2 Bug Fixes History 55
12 Legal Notices and 3rd Party Licenses 63

This version is not intended for GB/B customers. GB/B customers should use the designated release package to ensure full compatibility, qualification, and support alignment.

1 Overview

NVIDIA® Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)™ technology improves the performance of MPI and Machine Learning collective operation, by offloading collective operations from CPUs and GPUs to the network and eliminating the need to send data multiple times between endpoints.

This innovative approach decreases the amount of data traversing the network as aggregation nodes are reached, and dramatically reduces collective operations time. Implementing collective offloads communication algorithms supporting streaming for Machine Learning in the network also has additional benefits, such as freeing up valuable CPU and GPU resources for computation rather than using them to process communication.

With the 3rd generation of SHARP, multiple aggregation trees can be built over the same topology, enabling the aggregation and reductions benefits (also known as In-Network Computing) to many parallel jobs over the same infrastructure.

Further information on this product can be found in the following NVIDIA SHARP documents:

- Release Notes
- Deployment Guide

2 Document Revision History

For the list of changes made to this document, refer to Revision History.

3 Release Notes

Revision	Date	Description
3.15.0	June 2026	Initial release of this document version.

The release note pages provide the following information on NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP).

- General Information
- Changes and New Features
- Bug Fixes in this Version
- Known Issues

3.1 General Information

3.1.1 Packages Provided with SHARP

SHARP Software is provided with the following package:

Package	Version
HPC-X	2.50
UFM (Aggregation Manager only)	6.25
UFM Enterprise Appliance	1.15.1

3.1.2 Prerequisites

Device	Firmware/Software Version
NVIDIA Quantum	27.2014.2428
NVIDIA Quantum-2	31.2016.2054
NVIDIA Quantum-3[1]	35.2016.2016
ConnectX-6	20.44.1xxx
ConnectX-6 DE	22.49.1014
ConnectX-7	28.49.1014
ConnectX-8	40.49.1014

1. Support limited to NVIDIA Quantum-3 based XDR Q34xx-RA InfiniBand Switch.

3.1.3 NVIDIA Hardware Capabilities and Limitations

Device	Capabilities and Limitations
NVIDIA Quantum	• Supports both SHARP low latency and streaming aggregation operations. • Supports up to 126 aggregation trees in the subnet (63 low latency trees, and 63 streaming aggregation trees). **Note:** The number of SHARP streaming aggregation operations is limited to one active tree per switch.
NVIDIA Quantum-2	• Supports both SHARP low latency and streaming aggregation operations. • Supports up to 1023 aggregation trees in the subnet (511 low latency trees, and 511 streaming aggregation trees). **Note:** Multiple SHARP streaming aggregation operations can be operated in parallel by a single Quantum-2 switch. The limit is one active tree per port.
ConnectX-6 and above	Supports both SHARP low latency and streaming aggregation operations.

3.1.4 Supported Operating Systems and Platforms

For the complete list of supported Operating Systems and platforms, please refer to the list of operation systems and platforms supported by HPC-X.

3.2 Changes and New Features

3.2.1 Changes and New Features

Feature/Change	Description
Parallel SHARP All-Gather & Reduce-Scatter	[**Beta**] Introduced a high-performance **InfiniBand multicast** implementation for SHARP All-Gather (AG). This architectural shift allows SHARP AG to execute concurrently with SHARP Reduce-Scatter (RS), delivering significant throughput gains over traditional sequential execution.
Decoupling from DOCA-Host	Starting with this release, SHARP components are no longer bundled within the **DOCA-Host** installation package.
Bug Fixes	See Bug Fixes section.

3.2.2 Parameter Changes

N/A

3.3 Bug Fixes in this Version

Internal Reference Number	Issue
4993336	**Description**: Fixed an issue when exceeding 32 HCAs; increased the HCA limit to 128 specifically for Ubuntu 22.04.

Internal Reference Number	Issue
	Keywords: Virtual port; v-port; HCA; Ubuntu
	Discovered in Release: 3.10.3
	Fixed in Release: 3.15
4992952	**Description**: Fixed an issue where SHARP could not execute jobs using virtual ports on systems configured with non-default partition keys (pkeys).
	Keywords: pkeys; vport
	Discovered in Release: 3.14
	Fixed in Release: 3.15

3.4 Known Issues

Internal Reference Number	Issues
5037230	**Description**: During a UFM upgrade from a previous version, the `sharp_am` service may log an `"Invalid parameter stochastic_rounding_enabled"` error, causing the service to stop and automatically restart.
	Workaround: No action required. No impact on system performance or functionality; service resumes normal operation after the restart.
	Keywords: UFM; sharp_am; upgrade
	Discovered in Release: 3.15.0
-	**Description:** Using SHARP_am with switch firmware version 31.2014.3000 or later requires SHARP_am version 3.11.0 or newer.
	Workaround: N/A
	Keywords: SHARP_am; switch firmware
	Discovered in Release: 3.9.0
3340353	**Description:** When reconfiguring a standby management host to operate as a compute host, it will not be able to run SHARP jobs unless sharp_am is restarted. In case that a host runs the SM process, it will automatically be detected by the master SM as a standby SM and be reported as a standby management host. Note that restart is not required if ignore_sm_guids is set to FALSE.
	Workaround: N/A
	Keywords: active; standby; compute host; ignore_sm_guids
	Discovered in Release: 3.3.0
3371820	**Description:** Congestion Control cannot be configured on the same SLs used by sharp_am.
	Workaround: N/A
	Keywords: Congestion control; SL
	Discovered in Release: 3.3.0

Internal Reference Number	Issues
3305335	**Description:** When running mpirun with multiple groups, the following error message might be received: `[error] - AM QPAlloc confirm QP MAD response status 0x1c00` This message is received due to to the fact that multiple unserialized MAD requests are run in parallel.
	Workaround: Set the SHARP_COLL_SERIALIZE_MADS environment variable to TRUE when running mpirun.
	Keywords: mpirun; SHARP_COLL_SERIALIZE_MADS
	Discovered in Release: 3.2.0
3225401	**Description:** Dynamic trees creation feature does not support a case in which all root switches are down and restarted. If such a scenario takes place, sharp_am should be restarted once the root switches are up and running.
	Workaround: N/A
	Keywords: Aggregation Manager; sharp_am; dynamic trees
	Discovered in Release: 3.1.0
3237831	**Description:** SHARP does not support reassignment of LID values. In case LID reassignment is desired, make sure to stop all SHARP jobs, reassign LIDs via OpenSM, and restart sharp_am once the reassignment is done.
	Workaround: N/A
	Keywords: Aggregation Manager; OpenSM
	Discovered in Release: 3.1.0
3048427	**Description:** In the case that a switch split mode is modified (off/on), sharp_am does not handle the new number of supported ports unless it is restarted.
	Workaround: Restart sharp_am after changing a switch split mode definition.
	Keywords: Aggregation Manager; split mode
	Discovered in Release: 2.7.0
3051699	**Description:** Changing the configuration of SHARP switch ports using device_configuration_file does not take effect on disconnected split ports. If these ports are connected later, they will remain with their default configuration.
	Workaround: If the new configuration is desired for the split ports, make sure to restart the Aggregation Manager after connecting a split port to a host.
	Keywords: Aggregation Manager; split port
	Discovered in Release: 2.7.0
-	**Description**: On multi PKEY environment, UCX in SHARP can use only the default PKEY (PKEY at index 0).
	Workaround: Use sockets for communication over non-default PKEY.
	Keywords: Configuration, SMX, UCX, PKEY
	Discovered in Release: 2.4.3

Internal Reference Number	Issues
-	**Description:** High Availability for the Aggregation Manager is not supported in HPC-X/ DOCA-Host packages at this time. As a result, only one instance of the Aggregation Manager can operate within the InfiniBand fabric. When there is a handover or failover of the Subnet Manager, a new instance of the Aggregation Manager should be initiated on the host where the new Master Subnet Manager is active.
	Workaround: Use Aggregation Manager in UFM.
	Keywords: Aggregation Manager
-	**Description:** Aggregation manager should run on the same Host where the Master Subnet Manager (SM) is running.
	Workaround: N/A
	Keywords: Aggregation Manager
-	**Description:** Aggregation Manager should be started after completion of fabric configuration by the Subnet Manager.
	Workaround: N/A
	Keywords: Aggregation Manager
-	**Description:** Only Fat-Tree, Quasi-Fat-Tree, and Dragonfly+ topologies are supported by the Aggregation Manager.
	Workaround: N/A
	Keywords: Fabric Topology
-	**Description:** Only IB fabrics where all compute nodes are connected to NVIDIA SHARP capable switches are supported by the Aggregation Manager.
	Workaround: Manually configure mapping between the compute port and the Aggregation Node.
	Keywords: Fabric Topology
-	**Description:** Upon changes in configuration file beyond parameters in 3.3, Aggregation Manager should be restarted to deploy new configuration.
	Workaround: N/A
	Keywords: Configuration

4 Introduction

NVIDIA® Scalable Hierarchical Aggregation and Reduction Protocol (SHARP)™ technology improves the performance of MPI and Machine Learning collective operation, by offloading collective operations from CPUs and GPUs to the network and eliminating the need to send data multiple times between endpoints.

This innovative approach decreases the amount of data traversing the network as aggregation nodes are reached, and dramatically reduces collective operations time. Implementing collective offloads communication algorithms supporting streaming for Machine Learning in the network also has additional benefits, such as freeing up valuable CPU and GPU resources for computation rather than using them to process communication.

With the 3rd generation of SHARP, multiple aggregation trees can be built over the same topology, enabling the aggregation and reductions benefits (also known as In-Network Computing) to many parallel jobs over the same infrastructure.

5 NVIDIA SHARP Installation

NVIDIA SHARP consists of two main components: **sharp_am**, which runs in UFM, and **libsharp**, which is linked to client applications running on the compute nodes.

The recommended setup is to enable SHARP in UFM and install **libsharp** on the compute nodes using HPC-X.

5.1 Enabling SHARP in UFM

⚠ Starting UFM v6.23.0, **sharp_enabled** is set to **true** by default.

UFM Enterprise upgrade maintains the previous configuration. In this case, verify that the configuration file **conf/gv.cfg** includes the following setting:

```
[sharp]
sharp_enabled = true
```

When using the UFM Appliance, you can also run the command **lib sharp enable**, which applies the same configuration change.

⚠ In both UFM Enterprise and UFM Appliance, a UFM restart is required after making the change.

5.2 Installing the SHARP Client (libsharp) via HPC-X

The **libsharp** component (SHARP client) is installed on compute nodes using the HPC-X package.

Refer to the HPC-X User Manual for installation and configuration instructions, and ensure that HPC-X is installed in a shared directory accessible to all compute nodes.

For usage details after installation, see Using SHARP section.

To download the HPC-X packages, please visit here.

5.3 Testing the Setup

While **ibdiagnet** can be used to check for general fabric errors, it does not verify communication between **libsharp** and **sharp_am**, nor whether SHARP jobs can be successfully created.

To test the SHARP installation and configuration, two test applications are available in the **sharp/bin** directory. Both are designed to verify client-side behavior:

- **sharp_hello** – Verifies that a SHARP client can communicate with **sharp_am** and successfully create a SHARP job. The test creates a simple SHARP tree involving only one compute node. It checks job creation only and does not perform data aggregation.

- **sharp_coll_test** – Performs a more comprehensive test. It measures bandwidth between the node's HCA and its connected switch, and also verifies SHARP data transfer. This makes it a more thorough diagnostic than **sharp_hello**, as it checks both job creation and data aggregation.

5.3.1 Running Tests from the UFM Machine

Although it is recommended to run SHARP tests from a compute node, testing from the UFM machine is possible with a configuration change.

By default, **sharp_am** blocks **libsharp** from running on management nodes (including the active and standby UFM machines). To allow testing from the UFM node:

1. Edit the configuration file:
   conf/sharp/sharp_am.cfg
2. Set the following parameter:
   ignore_sm_guids = False
3. Restart **sharp_am** for the changes to take effect.

5.3.2 NVIDIA SHARP Hello

NVIDIA SHARP distribution provides sharp_hello test utility for testing SHARP's end-to-end functionality on a compute node. It creates a single SHARP job and sends a barrier request to SHARP Aggregation node.

Help

```
$sharp_hello -h
usage:  sharp_hello <-d | --ib_dev> <device> [OPTIONS]
OPTIONS:
        [-d | --ib_dev]      - HCA to use
        [-v | --verbose]     - libsharp coll verbosity level(default:2)
                                Levels: (0-fatal 1-err 2-warn 3-info 4-debug 5-trace)
        [-V | --version]     - print program version
        [-h | --help]        - show this usage
```

Example #1

```
$ sharp_hello -d mlx5_0:1 -v 3
[thor001:0:15042 - context.c:581] INFO job (ID: 12159720107860141553) resource request quota: ( osts:0
 user_data_per_ost:0 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[thor001:0:15042 - context.c:751] INFO tree_info: type:LLT tree idx:0 treeID:0x0 caps:0x6 quota: ( osts:167
 user_data_per_ost:1024 max_groups:167 max_qps:1 max_group_channels:1)
[thor001:0:15042 - comm.c:393] INFO [group#:0] group id:a tree idx:0 tree_type:LLT rail_idx:0 group size:1 quota:
(osts:2 user_data_per_ost:1024) mgid: (subnet prefix:0xff12a01bfe800000 interface id:0x3f020000000a) mlid:c007
Test Passed.
```

Example #2

```
$ SHARP_COLL_ENABLE_SAT=1 sharp_hello -d mlx5_0:1 -v 3

[swx-dgx01:0:59023 - context.c:581] INFO job (ID: 15134963379905498623) resource request quota: ( osts:0
 user_data_per_ost:0 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[swx-dgx01:0:59023 - context.c:751] INFO tree_info: type:LLT tree idx:0 treeID:0x0 caps:0x6 quota: ( osts:167
 user_data_per_ost:1024 max_groups:167 max_qps:1 max_group_channels:1)
[swx-dgx01:0:59023 - context.c:755] INFO tree_info: type:SAT tree idx:1 treeID:0x3f caps:0x16
[swx-dgx01:0:59023 - comm.c:393] INFO [group#:0] group id:3c tree idx:0 tree_type:LLT rail_idx:0 group size:1
 quota: (osts:2 user_data_per_ost:1024) mgid: (subnet prefix:0xff12a01bfe800000 interface id:0xd6060000003c)
mlid:c004
[swx-dgx01:0:59023 - comm.c:393] INFO [group#:1] group id:3c tree idx:1 tree_type:SAT rail_idx:0 group size:1
 quota: (osts:64 user_data_per_ost:0) mgid: (subnet prefix:0x0 interface id:0x0) mlid:0
```

```
Test Passed
```

6 Using SHARP

- NVIDIA SHARP Collective Library
- Using NVIDIA SHARP with Open MPI
- Using NVIDIA SHARP with NVIDIA NCCL
- Using NVIDIA SHARP with UCC

6.1 NVIDIA SHARP Collective Library

NVIDIA SHARP distribution provides a collective library implementation with high level API to easily integrate into other communication runtime stacks, such as MPI, NCCL and others.

The SHARP collective library offers collective operations such as Barrier, Allreduce, Reduce, Bcast, Reduce-scatter, and Allgather. It accommodates datatypes including 16/32/64-bit Integer/ Floating-point, as well as 16-bit Bfloat and 8-bit Integer.

6.1.1 NVIDIA SHARP Library Flags

6.1.1.1 NVIDIA SHARP Configuration Flags

As of NVIDIA SHARP version 2.7.0, sharpd daemon no longer exists, and its activity is now performed from application process.

The previous sharpd configuration is now done from the application command-line instead using the following flags.

Flag	Description
`SHARP_LOG_VERBOSTIRY`	Log verbosity level 1 - Errors 2 - Warnings 3 - Info 4 - Debug 5 - Trace **Default:** 2
`SHARP_LOG_FILE`	Log file **Default:** stdout The log file name accepts the following modifiers in the file name to create a unique file • **%D** date as DDMMYYYY • **%T** thread ID • **%H** host name
`SHARP_SMX_SOCK_INTERFACE`	Network interface to be used by SMX: empty string (default) - Use interface used for AM connection **Default**: (null)
`SHARP_SMX_SOCK_ADDR_FAMILY`	Determines which address family will be used in SMX's sockets. The value needs to be one of the following: { ipv4, ipv6, auto } **Default**: auto
`SHARP_SMX_UCX_INTERFACE`	Network interface to be used by SMX for UCX connections: empty string (default) - Use interface used for AM connection **Default**: (null)

6.1.1.2 NVIDIA SHARP Resource Tuning for Low Latency Operations

The following SHARP library flags can be used when running NVIDIA SHARP collectives.

Flag	Description
`SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST`	Maximum payload per OST (outstanding transactions). Value 0 means "allocate default value". **Valid values**: • 0 (default) • 128-1024
`SHARP_COLL_JOB_QUOTA_OSTS`	Maximum job (per tree) OST quota request. Value 0 means "allocate default quota". **Default**: 0
`SHARP_COLL_JOB_QUOTA_MAX_GROUPS`	Maximum number of groups (comms) quota request. Value 0 means "allocate default value". **Default**: 0
`SHARP_COLL_OSTS_PER_GROUP`	Number of OSTs per group. **Default**: 8
`SHARP_COLL_JOB_QUOTA_MAX_QPS_PER_PORT`	Maximum QPs/port quota request. Value 0 means "allocate default value".

6.1.1.3 NVIDIA SHARP Streaming Aggregation

The following NVIDIA SHARP library flags can be used to enable Streaming Aggregation Tree (SAT) and tuning.

Flag	Description
`SHARP_COLL_ENABLE_SAT`	Enables SAT capabilities. **Default**: 0 (Disabled) The Maximum message size SAT protocol support is 1073741792 Bytes (32B less than 1GB).
`SHARP_COLL_SAT_THRESHOLD`	Message size threshold to use SAT on generic allreduce collective requests. **Default**: 16384
`SHARP_COLL_SAT_LOCK_BATCH_SIZE`	SAT lock batch size. Set this to “1” if multiple communicators want to use SAT resources. **Valid range**: 1-65535. **Default**: 65535 (Infinity)
`SHARP_COLL_LOCK_ON_COMM_INIT`	Get SAT Lock resource during communicator init if lock batch size is Infinity. Return failure if failed to lock **Default:** 0 (Disabled), 1(Enabled) with NCCL SHARP plugin
`SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD`	Lazy group resource allocation. 0 - Disable lazy allocation, allocate group resource at communicator create time #n - Allocate sharp group resource after #n collective calls requested on the group **Default**: 1

Flag	Description
SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE	SAT (Streaming Aggregation Tree) exclusive lock mode for job. Possible values: • 0 - no exclusive lock • 1 - try exclusive lock • 2 (**default**)- force exclusive lock
SHARP_COLL_REDUCE_SCATTER_FRAG_SIZE	The fragment size of the reduce chunk in reduce-scatter collective operation. **Default** - 512M

6.1.1.4 SHARP Miscellaneous Tuning

Flag	Description
SHARP_COLL_ENABLE_CUDA	Enables CUDA GPU support. Possible values: • 0 - disable • 1 - enable • 2 **(default)** - try
SHARP_COLL_PIPELINE_DEPTH	Size of fragmentation pipeline for larger collective payload. **Default**: 64
SHARP_COLL_ENABLE_MCAST_TARGET	Enables MCAST target on NVIDIA SHARP collective operations. Possible values: • 0 (**default**) - disable • 1 - enable
SHARP_COLL_MCAST_TARGET_GROUP_SIZE_THRESHOLD	Group size threshold to enable mcast target. **Default**: 2
SHARP_COLL_POLL_BATCH	Defines the number of CQ completions to poll on at once. **Valid range:** 1-16 **Default**: 4
SHARP_COLL_ERROR_CHECK_INTERVAL	Interval in milliseconds that indicates the time between the error checks. If you set the interval as 0, error check is not performed. **Default**: 180,000
SHARP_COLL_JOB_NUM_TREES	Number of SHARP trees to request. 0 means requesting the number of trees based on the number of rails and the number of channels. **Default**: 0
SHARP_COLL_GROUPS_PER_COMM	Number of NVIDIA SHARP groups per user communicator. **Default**: 1
SHARP_COLL_JOB_PRIORITY	Job priority. **Valid values:** 0-10 **Default**: 0
SHARP_COLL_ENABLE_PCI_RELAXED_ORDERING	Enable PCI relaxed order memory access. Possible values: • 0 - disable • 1 - enable • 2 (**default**) - auto

⚠ For the complete list of SHARP_COLL tuning options, run the `sharp_coll_dump_config` utility:
`$HPCX_SHARP_DIR/bin/sharp_coll_dump_config`

6.2 Using NVIDIA SHARP with Open MPI

NVIDIA SHARP library is integrated into HCOLL collective library to offload collective operations in MPI applications.

The following basic flags should be used in environment to enable NVIDIA SHARP protocol in the HCOLL middleware. For the rest of flags, please refer to NVIDIA SHARP Release Notes.

6.2.1 HCOLL Library Flags

The following HCOLL flags can be used when running NVIDIA SHARP collective with mpirun utility.

Flag	Description
`HCOLL_ENABLE_SHARP`	Sets whether SHARP should be used. Possible values: • 0 (**default**) – do not use NVIDIA SHARP • 1 - probe NVIDIA SHARP availability and use it • 2 - force to use NVIDIA SHARP • 3 - force to use NVIDIA SHARP for all MPI communicators • 4 - force to use NVIDIA SHARP for all MPI communicators and for all supported collectives (barrier, allreduce)
`SHARP_COLL_LOG_LEVEL`	NVIDIA SHARP coll logging level. Messages with a higher or equal level to the selected will be printed. Possible values: • 0 - fatal • 1 - error • 2 (**default**) - warn • 3 - info • 4 - debug • 5 - trace
`HCOLL_SHARP_NP`	Number of nodes (node leaders) threshold in the communicator to create NVIDIA SHARP group and use NVIDIA SHARP collectives. **Default**: 4
`HCOLL_SHARP_UPROGRESS_NUM_POLLS`	Number of unsuccessful polling loops in libsharp coll for blocking collective wait before calling user progress (HCOLL, OMPI). **Default**: 999
`HCOLL_ALLREDUCE_SHARP_MAX` (or) `HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX`	Maximum allreduce size run through NVIDIA SHARP. A message size greater than the above the specified value by this parameter will fall back to non-SHARP-based algorithms (multicast based or non-multicast based). The threshold is calculated based on the group resources. Threshold = #OSTS * Payload_per_ost **Default**: Dynamic

6.2.1.1 Example of Allreduce with Default Settings with SHARP Enable

```
$ mpirun -np 128 -map-by ppr:1:node  -x UCX_TLS=dc,shm,self  -x HCOLL_ENABLE_SHARP=3 -x
SHARP_COLL_ENABLE_SAT=1  $HPCX_OSU_DIR/osu_allreduce

# OSU MPI Allreduce Latency Test v5.6.2
# Size       Avg Latency(us)
4                       7.44
8                       8.43
16                      7.81
32                      8.55
64                      9.06
128                     8.44
256                     9.41
512                     8.50
1024                    9.03
2048                   10.43
4096                   42.61
8192                   37.93
16384                  15.48
32768                  16.26
65536                  17.62
131072                 23.09
262144                 33.90
524288                 58.98
1048576               101.53
```

6.3 Using NVIDIA SHARP with NVIDIA NCCL

RDMA and SHARP collectives are enabled with NVIDIA NCCL ('nickel') collective communication library through the NCCL-SHARP plugin.

The NCCL-SHARP plugin is distributed through the following channels:

- Binary distribution with HPC-X. The plugin will be loaded in the environment with HPC-X modules and NCCL will load it automatically. The plugin can be built from the source of other CUDA versions.
- Source distribution: https://github.com/Mellanox/nccl-rdma-sharp-plugins
  User can build the plugin from the source and set LD_LIBRARY_PATH to use it by NCCL.

6.3.1 Requirements

- NVIDIA ConnectX-6 HDR and above
- NVIDIA Quantum HDR switch and above
- DOCA-Host
- GPUDirectRDMA
  - Plugin
  - Source code repository

  It is important to verify that the GPUDirect RDMA kernel module is properly loaded on each of the computing systems where you plan to run the job that requires the GPUDirect RDMA.

  To check whether the GPUDirect RDMA module is loaded, run:

  ```
  # service nv_peer_mem status
  ```

  To run this verification on other Linux flavors:

  ```
  # lsmod | grep nv_peer_mem
  ```

- NCCL version 2.7.3 or higher
  Please refer to NVDIA's Developer Guide for more details: https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/index.html

6.3.2 Control Flags

The following environment variables enable the SHARP aggregation with NCCL when using the NCCL-SHARP plugin.

- NCCL variables:
  - `NCCL_COLLNET_ENABLE=1`
  - `NCCL_ALGO=CollNet` (Required to overcome a bug in NCCL <= 2.7.8)
- SHARP variables:
  - For guaranteed SAT resources on initialization: These options are enabled by default with NCCL SHARP Plugin version >= 2.1.x. Users can enable explicitly using following variables:
    - SHARP_COLL_LOCK_ON_COMM_INIT=1 (
    - SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD=0
  - [Optional] SHARP_COLL_LOG_LEVEL=3
- NCCL SHARP Plugin variables:
  - NCCL_SHARP_DISABLE
    - NCCL SHARP Streaming aggregation is supported on a single NCCL communicator/process group (PG). Applications can selectively enable SHARP on specific Process Group (PG) by setting this variable in the application before creating the PG
  - NCCL_SHARP_GROUP_SIZE_THRESH
    - Application can set this code option to selectively enable SHARP on the PG based on the group size
  - NCCL_IBEXT_DISABLE
    - NCCL plugin will be disabled and NCCL native communication transports will be used instead

6.3.3 Cluster Topology for Using NVIDIA SHARP SAT with NCCL

On systems with multiple GPUs and multiple HCAs, NCCL creates an aggregation streaming flow (NCCL Ring/Channel) per HCA rail. It is required to build the cluster topology in such a way that leaf level switches connected to same HCA rail from each server.

6.3.4 NCCL Benchmark Example

The sanity performance of the setup can be verified with NCCL tests. Please refer to NCCL tests here: https://github.com/NVIDIA/nccl-tests

Example:

```
$ mpirun  -np 1024  -map-by ppr:8:node -x UCX_TLS=dc,shm,self   -x LD_LIBRARY_PATH=/sw/nccl/build/lib::/sw/nccl-
rdma-sharp-plugins/install/lib:$LD_LIBRARY_PATH -x  NCCL_COLLNET_ENABLE=1  all_reduce_perf -b 4 -e 2G -f 2 -g 1 -w
50 -n 50

           4             1   float     sum    44.53    0.00    0.00  3e-05    44.21    0.00    0.00  3e-05
           8             2   float     sum    45.42    0.00    0.00  3e-05    45.85    0.00    0.00  3e-05
          16             4   float     sum    46.34    0.00    0.00  3e-05    45.84    0.00    0.00  2e-05
          32             8   float     sum    46.20    0.00    0.00  2e-05    46.56    0.00    0.00  2e-05
          64            16   float     sum    46.00    0.00    0.00  2e-05    48.33    0.00    0.00  2e-05
         128            32   float     sum    48.77    0.00    0.01  2e-05    47.23    0.00    0.01  2e-05
         256            64   float     sum    47.88    0.01    0.01  2e-05    47.85    0.01    0.01  2e-05
         512           128   float     sum    51.44    0.01    0.02  3e-05    48.66    0.01    0.02  3e-05
        1024           256   float     sum    51.27    0.02    0.04  4e-05    51.78    0.02    0.04  4e-05
        2048           512   float     sum    57.93    0.04    0.07  4e-05    56.45    0.04    0.07  4e-05
        4096          1024   float     sum    57.32    0.07    0.14  4e-05    93.51    0.04    0.09  4e-05
        8192          2048   float     sum    106.4    0.08    0.15  4e-05    59.70    0.14    0.27  4e-05
       16384          4096   float     sum    103.0    0.16    0.32  4e-05    58.23    0.28    0.56  4e-05
       32768          8192   float     sum    74.85    0.44    0.87  4e-05    137.8    0.24    0.48  4e-05
       65536         16384   float     sum    96.71    0.68    1.35  4e-05    92.89    0.71    1.41  4e-05
      131072         32768   float     sum    115.6    1.13    2.27  4e-05    120.7    1.09    2.17  4e-05
      262144         65536   float     sum    197.7    1.33    2.65  4e-05    167.6    1.56    3.13  4e-05
      524288        131072   float     sum    222.7    2.35    4.70  4e-05    239.2    2.19    4.38  4e-05
     1048576        262144   float     sum    280.9    3.73    7.46  4e-05    197.7    5.30   10.60  4e-05
     2097152        524288   float     sum    218.0    9.62   19.22  4e-05    213.9    9.81   19.59  4e-05
     4194304       1048576   float     sum    257.6   16.28   32.53  4e-05    254.7   16.47   32.90  4e-05
     8388608       2097152   float     sum    354.3   23.68   47.31  4e-05    523.5   16.02   32.02  4e-05
    16777216       4194304   float     sum    505.9   33.16   66.26  4e-05    484.1   34.66   69.24  4e-05
    33554432       8388608   float     sum    639.2   52.50  104.89  4e-05    678.6   49.45   98.80  4e-05
    67108864      16777216   float     sum   1358.2   49.41   98.72  4e-05   1048.6   64.00  127.87  4e-05
   134217728      33554432   float     sum   1737.2   77.26  154.37  4e-05   1777.6   75.51  150.86  4e-05
   268435456      67108864   float     sum   4359.5   61.58  123.03  4e-05   4262.3   62.98  125.83  4e-05
   536870912     134217728   float     sum   5619.7   95.53  190.88  4e-05   5699.0   94.20  188.22  4e-05
  1073741824     268435456   float     sum    12169   88.23  176.30  4e-05    11508   93.30  186.42  4e-05
  2147483648     536870912   float     sum    22618   94.94  189.70  4e-05    21814   98.44  196.70  4e-05
# Out of bounds values : 0 OK
# Avg bus bandwidth    : 41.2497
#
```

6.4 Using NVIDIA SHARP with UCC

NVIDIA SHARP library is integrated into the Unified Collective Communication (UCC) library to offload collective operations in MPI applications.

The following flags should be used in the environment to enable the NVIDIA SHARP protocol in the UCC middleware.

6.4.1 UCC Library Flags

The following UCC flags can be used when running NVIDIA SHARP collective with mpirun utility.

Flag	Description
OMPI_UCC_CL_BASIC_TLS	Currently, the UCC supported/compiled TLs list is: ucp,cuda,nccl,sharp **Note**: By default, "sharp" is disabled. Possible values: ucp,sharp: Adding "sharp" TL along with "ucp" ucp,cuda,sharp: Adding "sharp" TL along with "ucp" and "cuda"
UCC_TL_SHARP_MIN_TEAM_SIZE	Minimal UCC team size for which sharp can be used. Default: 2
UCC_TL_SHARP_DEVICES	List of comma-separated HCAs to be used with SHARP TL.
UCC_TL_SHARP_UPROGRESS_NUM_POLLS	Number of unsuccessful polling loops in libsharp coll for blocking collective wait before calling user progress (UCC, OMPI). Default: 999
UCC_TL_SHARP_ENABLE_LAZY_GROUP_ALLOC	Enables lazy group resource allocation Default: N

6.4.1.1 Example of Allreduce with Default Settings with SHARP Enable

```
$mpirun  --bind-to core --map-by node -np 8  -x LD_LIBRARY_PATH -x SHARP_COLL_LOG_LEVEL=3 -x
SHARP_COLL_ENABLE_SAT=1 -x   OMPI_UCC_CL_BASIC_TLS=ucp,sharp  $HPCX_OSU_DIR/osu_allreduce
[elsa01:0:1852929 - context.c:687][2024-10-28 22:24:47] INFO job (ID: 139896884886908) resource request quota:
( osts:0 user_data_per_ost:0 max_groups:0 max_qps:1 max_group_channels:1, num_trees:1)
[elsa01:0:1852929 - context.c:889][2024-10-28 22:24:47] INFO sharp_job_id:3    resv_key: tree_type:LLT tree_idx:0
  treeID:2 caps:0x66 quota:(osts:23 user_data_per_ost:1024 max_groups:23 max_qps:1 max_group_channels:1)
[elsa01:0:1852929 - context.c:896][2024-10-28 22:24:47] INFO sharp_job_id:3    tree_type:SAT tree_idx:1  treeID:514
 caps:0x76
[elsa01:0:1852929 - comm.c:413][2024-10-28 22:24:47] INFO [group#:0] job_id:3 group id:0 tree idx:0 tree_type:LLT
rail_idx:0 group size:6 quota: (osts:8 user_data_per_ost:1024) mgid: (subnet prefix:0x0 interface id:0x0) mlid:0
[elsa01:0:1852929 - comm.c:413][2024-10-28 22:24:47] INFO [group#:1] job_id:3 group id:0 tree idx:1 tree_type:SAT
rail_idx:0 group size:6 quota: (osts:64 user_data_per_ost:0) mgid: (subnet prefix:0x0 interface id:0x0) mlid:0



# OSU MPI Allreduce Latency Test v7.4
# Datatype: MPI_FLOAT.
# Size       Avg Latency(us)
4                       4.95
8                       4.80
16                      5.08
32                      5.10
64                      5.62
128                     5.74
256                     7.27
512                     7.85
1024                    8.28
2048                    8.59
4096                   14.64
8192                   11.11
16384                  10.31
32768                  11.42
65536                  13.84
131072                 20.07
262144                 37.93
524288                 33.89
1048576                64.64
```

7 SHARP in Public Cloud

Deploying SHARP in a public cloud environment requires special considerations to ensure fair resource usage across tenants and to prevent one tenant from disrupting another.

Two key measures should be taken:

1. Setting a Secret AM Key
   This ensures that only **sharp_am** is authorized to perform configuration changes. Without this safeguard, a tenant's application could potentially impersonate **sharp_am** and alter fabric settings.
2. Using PKeys
   PKeys are used (independently of SHARP) to isolate traffic between tenants. SHARP automatically integrates with this mechanism to enforce resource separation and access control aligned with the PKey setup.

7.1 Secret AM Key

The Secret AM Key is a 64-bit value that must be defined by the cloud administrator and set in the **sharp_am** configuration file.

This key must remain strictly confidential and must not be shared with or exposed to any cloud tenant.

Once configured, **sharp_am** programs all connected switches with this key. From that point forward, switches will only accept Sharp-related MADs that include the correct Secret AM Key.

MADs originating from **libsharp** use a separate key known as the *Sharp Job Key*. This key is dynamically generated per job and distributed by **sharp_am** to the corresponding **libsharp** instance, ensuring isolation between tenants and preventing one tenant from sending MADs on behalf of another.

If a MAD is received by a switch with an incorrect key, it is silently dropped, and the switch emits an **AMKeyViolation** trap (trap number 257) to **sharp_am** . Cloud administrators should monitor event logs for such traps. A high volume of these traps may indicate a brute-force attempt by a tenant to discover the key.

7.2 SHARP PKeys Support

Sharp can be configured to automatically treat the PKeys definitions as the system tenants.

This ensures that tenants can use SHARP without interfering with one another's, and gives administrators fine-grained control over SHARP usage.

To enable SHARP in PKeys support mode, the following prerequisites must be met:

- **sharp_am** must be running inside UFM.
- In the UFM configuration file **gv.cfg** , set:

```
enable_sharp_allocation = True
```

Once PKeys support mode is enabled, compute nodes are not permitted to initiate SHARP jobs unless they belong to a defined PKey.

The system admin can control which PKeys are entitled to use SHARP and which are not. By Default, every defined PKey can use SHARP. The PKey API provides an optional field to mention whether SHARP can be used by applications that belong to the PKey.

The PKeys definitions are created and managed via UFM GUI and REST API, which supports:

- Creating, updating, and deleting PKeys.
- Defining whether a PKey should be able to use Sharp or not.

This mechanism gives fabric administrators the ability to define per-tenant SHARP entitlements and enforce strict isolation.

For full details, refer to the NVIDIA UFM Enterprise REST API Guide.

7.2.1 SHARP Resource Limits

Enforcing SHARP resource limits per tenant is essential in multi-tenant environments to maintain fairness and avoid resource contention.

By default:

1. A tenant can run multiple SHARP jobs concurrently.
2. No two SHARP jobs may share the same HCA.
3. There is no global limit on the number of jobs a tenant may launch. However, since each SHARP job requires at least 2 HCAs, and each HCA may only serve one job, the effective job limit per tenant is approximately half the number of available HCAs.

These default constraints ensure that in a non-blocking topology, no tenant can monopolize resources or degrade performance for others.

Administrators can override defaults by:

- Setting a global job limit for all tenants via a configuration parameter.
- Adjusting the number of SHARP jobs allowed per HCA, via a global configuration parameter.

These controls provide the flexibility to tailor SHARP behavior to specific cloud tenancy models and fairness policies.

7.3 Limitation

SHARP can attach a compute node HCA to only one PKey at any given time.

A compute node HCA can have full PKey membership in multiple PKeys, but SHARP will attach it to only one of them for SHARP Jobs.

If an HCA has full membership in multiple PKeys, SHARP will arbitrarily select one PKey to attach it to, and will generate warning events about it.

In multi-tenant cloud environments, such configurations are uncomoon, as HCAs typically have full membership in only one PKey.

8 SHARP Monitoring

- sharp_am Log and Dump Files
- sharp_am Telemetry
- NVIDIA SHARP Traps
- Aggregation Trees Diagnostics

8.1 sharp_am Log and Dump Files

The **sharp_am** logs its active logs to a log file named **sharp_am.log**.

sharp_am also generates various dump files, useful for monitoring and used by **sharp_am** at restart, to retain the previous run state.

Since some of the dump files are used by **sharp_am** at restart, it is important not to modify their content.

8.1.1 Activity Log Verbosity Level

Configuration parameters allow control over the verbosity level of the activity log file. The logged messages are categorized by relevancy, such as those related to network activity or SHARP trees calculations. Each category can have a distinct verbosity level. The two configuration parameters that control the log verbosity are **log_verbosity** and **log_categories_file**.

The **log_verbosity** config parameter functions as the main log verbosity parameter. The value set in this parameter defines the desired log verbosity for all categories unless specified differently for a particular category.

The **log_categories_file** parameter specifies the full path to a configuration file that defines the desired log verbosity for each category. By default in UFM, the provided file does not set specific levels to any category; all categories are commented out.

If a category is defined in the **fabric_log_catgeories.cfg** file, its definition overrides the main log verbosity. **Sharp_am log** verbosity can be updated without restarting by sending a SIGHUP signal. When updating the **sharp_am** configuration, you can modify the main log verbosity, update the location of the categories file, and adjust the content of the categories file.

8.1.2 Log Levels

There are five log levels, and their configuration is determined by numerical representation, however, the log messages are displayed with their full names.

The log levels include:

1. Error
2. Warning
3. Info
4. Debug
5. Trace/Verbose

When operating under normal conditions, it is advisable to set the log level to 3-Info.

8.1.3 Log Categories Config File

The configuration file included in the SHARP package lists possible categories but does not assign any values to them initially, as they are all commented out. To adjust the log level for a specific category, it is necessary to remove the "#" symbol and set the desired log level.

To implement the modification, either restart **sharp_am** or send a SIGHUP signal.

The provided package file contains the following text:

```
# A line starting with "#" is a comment line
# To set a specific category, remove the "#"
#
# Possible levels are: 1-5
# 1 - Error, 2-Warning, 3-Info, 4-Debug, 5-Verbose

#SHARP_GENERAL = 3
#SHARP_SR = 3
#SHARP_SMX = 3
#SHARP_SIGNAL = 3
#SHARP_MADS = 3
#SHARP_JOBS = 3
#SHARP_RESERVE = 3
#SHARP_FGRAPH = 3
#SHARP_FABRIC = 3
#SHARP_TREES = 3
#SHARP_RDMA_SR = 3
```

8.2 sharp_am Telemetry

When using Unified Fabric Manager (UFM), sharp_am publishes statistical data, accessible through an HTTP endpoint in CSV, Prometheus, or JSON formats.

sharp_am generates this data at consistent intervals (recommended: every 60 seconds), regardless of whether it is being actively requested. Because of this, frequent polling will return the same data, so it's advisable to retrieve information at intervals similar to those configured for sharp_am data updates.

The published data fields include the following:

Field Name	Description
metadata_host	Hostname of the server running sharp_am.
metadata_timestamp	Unix timestamp (in seconds) indicating when data was generated; independent of request time.
timestamp	Unix timestamp (in milliseconds) showing when data was requested.
active_jobs	Total number of currently active SHARP jobs.
active_sat_jobs	Active SHARP jobs specifically requesting SAT rather than just LLT.
agg_nodes_in_invalid_state	Aggregation nodes (switches) in an invalid state and excluded from resource allocation.

The data includes histogram fields, such as **active_jobs_num_hcas_histogram_bucket_X**, representing active jobs based on the number of HCAs each job serves. Each bucket corresponds to a range of HCAs, with the bucket labeled **_infinity** covering jobs with 1025 or more HCAs.

Similarly, **trees_level_histogram_bucket_X** fields provide a histogram of active jobs by SHARP tree level. For instance, a job using HCAs connected to the same leaf switch (requiring only one level) would be counted in **trees_level_histogram_bucket_0**.

Historical Data Fields

In addition to current metrics, sharp_am also provides historical statistics:

Field Name	Description
history_starting_timestamp	Start time for historical data collection, which resets on restart or failover.
history_denied_reservations	Count of denied reservation requests, which may indicate configuration issues.
history_denied_jobs_by_reservations	Count of job requests denied due to mismatched reservations.
history_denied_jobs_by_resource_limit	Count of job denials due to insufficient resources, potentially due to disconnected or invalid switches.
history_jobs_ended_due_to_client_failure	Number of jobs that ended due to client-side failure.
history_jobs_ended_due_to_fatal_sharp_error	Number of jobs that ended due to switch failure or link error.
history_jobs_ended_successfully	Number of jobs completed without issues.
history_ended_jobs_duration_in_hours_histogram_bucket_X	Job durations (in hours) of completed jobs, segmented into histogram buckets.

For example, a job active for less than one hour would fall under **history_ended_jobs_duration_in_hours_histogram_bucket_1**, while one running for six days would be counted in **history_ended_jobs_duration_in_hours_histogram_bucket_168**.

Fetching Data

To retrieve this data, use port 9002 (if configured as default) and one of the following endpoints:

Endpoint	Response Format
/csv/fset/sharp_am	CSV format
/json/fset/sharp_am	JSON format
/fset/sharp_am	Prometheus format

Example of a JSON data request:

```
curl --silent http://localhost:9002/json/fset/sharp_am | jq
```

Enabling UFM Configuration

NVIDIA SHARP telemetry is disabled by default. To enable it:

1. Edit the file **/opt/ufm/conf/secondary_telemetry_defaults/launch_ibdiagnet_config.ini**
2. Set the following parameter to 0:

```
plugin_env_CLX_EXPORT_API_DISABLE_SHARP_AM=0
```

3. Restart the ufm telemetry service for the changes to take effect:

```
/etc/init.d/ufmd ufm_telemetry_restart
```

8.3 NVIDIA SHARP Traps

Sharp traps are notifications sent by the switches to sharp_am. The traps alert sharp_am about various sharp-related events. Some traps can indicate a possible error in the client application logic, some traps can indicate a potential error in the Switch or in the Sharp software and require investigation by Nvidia experts.

All the traps received by sharp_am are displayed in sharp_am log file and in UFM events.

All traps are reported with the relevant switch LID. Some traps provide additional information, such as the relevant job, relevant QPs and a syndrome that gives a more precise reason for the trap.

In many cases when traps are sent to sharp_am, an error is also detected by the client app, and the following error is displayed in the client log file/console: "ERROR SHArP Error detected. err code:<> type:<> desc: <>".

The following table lists all the Sharp Traps, and their properties.

Trap Name	Trap Number	Description	Possible Root Cause
AMKey Violation	257	Tells that someone tried to send a MAD with wrong AM-KEY. This is a security warning, libsharp is using the right AM-KEY and Job-Key the sender of the MADs should be checked.	Check for unauthorized app in the system. Ina public cloud system, it can suggest one of the tenants is trying to hack the system.
QPError	132	The TRAP means that the QP has entered an error state. There are 2 syndromes: 3 - means that the QP has received a higher amount of RNR than it was configured to allow. 4 - The QP didn't receive an ACK on a packet, and it has reached the maximum amount of 'retries' that was configured for it.	This trap can occur when there is a physical link issue.
QPAllocation-Timeout	133	This trap tells that a QP Allocation request was received by the switch, but a QP Allocation Confirmation was not received by the switch afterward, and the timeout expired for waiting for the confirmation.	In case the client application terminated abnormally during the initialization phase, there is a slight probability that the application managed to send the QP Allocation MAD and terminated before sending the Confirmation MAD.

Trap Name	Trap Number	Description	Possible Root Cause
SharpInvalidRequest	134	This trap tells that there is an error while trying to aggregate the received data. Syndromes: 0 - Quota Violation: Occurs when a job exceeds its allocated quota limits, indicates resource allocation violation in the SHARP protocol. 1 - Invalid Version: Indicates that the SHARP protocol version in the request is not supported, used for protocol version compatibility checking. 2- Invalid Opcode: Indicates that the operation code in the request is not valid, used to validate SHARP operation types. 3 - Invalid Vector/Payload Size: Indicates that there is a mismatch between the header information and the actual payload size. 4 - Job ID Violation: Occurs when there's a mismatch between the job ID in the QPC and the header tuple. 5 - Invalid Tree ID: Indicates that the specified tree is not active, used for tree state validation. 6 - Tree Violation: Indicates that there is a mismatch between the tree ID in the QPC and the header tuple. 7 - OST Mismatch: Indicates that at least two packets in a transaction have mismatch in their properties (For example: not the same payload size). 8 - Child Not In Group: Occurs when a request comes from a non-member of the group, used for group membership validation. 9 - Bad Target Header: Indicates an invalid target header in the request. 10 - Job Lock Violation: SAT lock for a job failed, because the port semaphore is already locked for another job. 11 - SAT Operation Without Lock: Occurs when a SAT operation is received without proper lock, ensures proper locking sequence. 12 - Unsupported Job ID: Job ID is above the maximal allowed number. 13 - Bad SAT Lock Operation: Indicates that a job tries to lock the same semaphore twice (can take place due to SLB packets retries). 14 - (LLT Only) Sharp Payload Not Aligned: Indicates that the SHARP payload is not properly aligned, used for memory alignment validation. 15 - ANDR Request On Non SAT: Occurs when an ANDR request is made on a non-SAT operation, used for operation type validation. 16 - Group Context Does Not Exist: Indicates that the requested group context does not exist, used for group context validation.	Depending on the syndrome, this error can be a result of a wrong logic of the client application or an internal Sharp issue. The following syndromes hint to a problem in the client app: 2 - Invalid Opcode: The different ranks of the application are not using the same aggregation/reduction logic. 17 - Bad SAT Unlock Operation: Can occur when client application terminates abnormally, and there are still pending operations related to this application. All the other syndromes, hint to an issue in Sharp logic.

Trap Name	Trap Number	Description	Possible Root Cause
		17 - Bad SAT Unlock Operation: Indicates that a SAT unlock operation failed, because the switch is still handling packets related to the job. 18 - No Free Buffers: Indicates that there are no available memory buffers. Can hint to a memory leak. 255 - General Error: Triggered by default case in error handling, used when the error cannot be associated with any specific syndrome.	
SharpError	135	Reports streaming aggregation errors (SAT). This trap focuses on streaming aggregation issues. Syndromes: 0 - Lock Semaphore Timeout: A lock was acquired, but was not used before timeout expired. 1 - OST No Progress Timeout: Started to receive a message, but not all packets of the message arrived before timeout expired. 2 - Bad SAT Request Classification: A transaction arrived with a request to perform an operation that is not supported by SAT. 3 - SAT ANDR Got Multiple Targets: Ambiguous routing for Reduce Scatter. The reduce operation has more than 1 target, but should have only 1. 4 - SAT Bad Target Header: SAT operation with an invalid SAT properties in the Target Header. 5 - SAT No Destination On Root: ANDR request with no destination on root switch. 6 - SAT Exceeds Number Of Outstanding Operations: Max number of transactions 'in the air' per tuple is 64, and this limit is breached. 7 - SAT Unbalanced FIFO Data: Unbalanced payload issue. 8 - SAT Unbalanced OST Address: Out-of-sequence issue. 9 - SAT Data Corruption: Data integrity issue. 10 - SAT Bad Data Granularity: Data size issue.	Depending on the syndrome, this error can be a result of a wrong logic of the client application or an internal Sharp issue. The following syndromes hint to a problem in the client app: 1 - OST No Progress Timeout: Can occur due to physical link issues, or when a client application stalls for a long time while transmitting data, for example getting no CPU resources due to high CPU load of another application. 7 - SAT Unbalanced FIFO Data: The different ranks of the application are not using the same aggregation/reduction length. 10 - SAT Bad Data Granularity - This can take place if the application sent truncated data. All the other syndromes, hint to an issue in Sharp logic.

8.4 Aggregation Trees Diagnostics

Run *ibdiagnet* utility with SHARP diagnostics option.

```
$ibdiagnet --sharp
```

Check *fabric summary* table in *ibdiagnet* output for the number of identified aggregation nodes. For example:

```
Fabric Summary

Total Nodes            : 24
IB Switches            : 4
IB Channel Adapters    : 16
IB Aggregation Nodes   : 4
IB Routers             : 0

Total number of links  : 24
Links at 4x50          : 24

Master SM: Port=1 LID=1 GUID=0x248a070300a28c4d devid=4119 Priority:0 Node_Type=CA Node_Description=pnemo HCA-2
Standby SM : No Standby SM
```

Check *summary* table in *ibdiagnet* output for errors in SHARP diagnostics stage. For example:

```
Summary
-I- Stage                    Warnings   Errors     Comment
-I- Discovery                0          0
-I- Lids Check               0          0
-I- Links Check              0          0
-I- Subnet Manager           0          0
-I- Port Counters            0          0
-I- Nodes Information        0          0
-I- Speed / Width checks     0          0
-I- Alias GUIDs              0          0
-I- Virtualization           0          0
-I- Partition Keys           0          0
-I- Temperature Sensing      0          0
-I- SHARP                    0          0
```

Check in SHARP diagnostics output file (/var/tmp/ibdiagnet2/ibdiagnet2.sharp) that SHARP aggregation trees are configured in the subnet.

For example: count number of configured aggregation trees constructed by Aggregation Manager using *grep* command:

```
$cat /var/tmp/ibdiagnet2/ibdiagnet2.sharp | grep -c TreeID
126
```

Note that when operating in dynamic trees mode, ibdiagnet may print warning messages about the existence of multiple distinct trees with the same tree ID. In dynamic trees mode, this is a valid situation and these warnings should be ignored.

Warning example:

```
-W- <> - In Node <> found root tree (parent qpn <>) which is already exists for treeID: <>
```

9 Appendices

- Appendix A: Using SHARP with Quasi Fat Trees and DFP Topologies
- Appendix B: SHARP_am Network Interfaces
- Appendix C: HCA-to-TOR Benchmark Tool
- Appendix D: Stochastic Rounding Support
- Appendix E: NVIDIA SHARP Cleanup
- Appendix F: Running SHARP_am from HPC-X
- Appendix G: Disabling SHARP on Specific Network Devices in OpenSM

9.1 Appendix A: Using SHARP with Quasi Fat Trees and DFP Topologies

sharp_am considers multiple factors when determining how to allocate trees for each job. While the trees must satisfy the job's requirements, **sharp_am** also aims to preserve fabric resources - such as links and switches - for future jobs.

Topology plays a key role in how trees are allocated. The available combinations differ significantly between standard Fat-Tree, Quasi-Fat Tree (QFT), and Dragonfly (DFP) topologies. Accordingly, **sharp_am** supports two distinct allocation algorithms:

- Fat-Tree Algorithm (default) – Optimized for standard Fat-Tree fabrics.
- QFT/Dragonfly Algorithm – Tailored for Quasi-Fat Tree and Dragonfly topologies.

To use the QFT/Dragonfly algorithm, update the configuration file **conf/sharp/sharp_am.cfg** and set:

```
dynamic_tree_algorithm = 1
```

A restart of **sharp_am** is required for the change to take effect.

Notes:

- Only one algorithm can be active at a time.
- Restart **sharp_am** when switching algorithms.
- In some scenarios, using one algorithm with concurrent jobs may result in a “No resources” error for a specific job. Switching to the alternative algorithm may resolve the issue by redistributing resources differently.

Fabric-Specific Behavior

When operating in QFP fabric, use of the QFP-optimized algorithm is mandatory. In fact, **sharp_am** will automatically enforce this mode, regardless of the configured value.

In Fat-Tree or Quasi-Fat Tree topologies, using the corresponding optimized algorithm is optional —but recommended. In certain cases, NVIDIA may advise using the non-default algorithm based on specific workload or topology characteristics.

Algorithm Selection Support

Selecting the appropriate algorithm can significantly impact SHARP efficiency and job success. For guidance tailored to your system, contact NVIDIA Enterprise Support:

- Email: Enterprisesupport@nvidia.com
- Web: https://www.nvidia.com/en-us/support/enterprise

9.2 Appendix B: SHARP_am Network Interfaces

SHARP_am communicates with the following entities:

- IB switches - SHARP_am sends MADs to get status and configure the switches for SHARP activities.
  The MADs communication with IB switches takes place over the IB network.
- libsharp - Rank0 of collective operation, sending SHARP job requests to SHARP_am and receiving SHARP_am instructions.
  The communication with libsharp is performed via a proprietary binary protocol called smx.
  The transport layer of the smx can be via IB using UCX (InfiniBand transport), or via sockets (Ethernet).
- UFM - when operating inside UFM, various information and configuration commands are passed from UFM to SHARP_am.
  The communication with UFM is also performed via the smx proprietary protocol. However, the transport layer of this communication is unix-socket.

9.2.1 Network Interfaces Configuration

By default, SHARP_am uses the opensm IB interface for the MADs and libsharp communication.

The communication with libsharp is done via socket (Ethernet) transport by default.

A unix-socket is always kept open for communication with UFM.

It is possible to modify the communication protocol and control various network settings, using the following configuration parameters:

Parameter	Component	Description
`ib_port_guid`	`SHARP_am`	Sets the GUID of the port to which SHARP_am binds to, for all MAD communication with the switches. Value of 0 means to use the same port that is used by OpenSM. Default value: 0
`smx_protocol`	`SHARP_am`	Defines the default protocol that will be used when communicating with libsharp. Value 1 - UCX. Value 2 - Sockets. Default value: 2, which means sockets.

Parameter	Component	Description
smx_sock_interface	SHARP_am	Relevant only in case that smx socket transport is enabled. Sets the interface to be used by smx for the sockets connections. The interface should be mentioned by its name. Empty value means to use the same interface used by OpenSM, using IP-over-IB in this case. When SHARP_am is operating inside UFM, this parameter is automatically set by UFM according to its internal logic and the am_interface parameter in the gv.cfg file. Default value: Empty.
smx_sock_port	SHARP_am	Relevant only in case that smx socket transport is enabled. IP port number to be used for the socket listener. Default value: 6126
smx_sock_addr_family	SHARP_am	Relevant only in case that smx socket transport is enabled. Determines which address family will be used in SMX's sockets. IPv4, IPv6 or try both. A value 'auto' will use both IPv4 and IPv6 if they are available. Default value: auto
smx_ucx_interface	SHARP_am	Relevant only in case that smx UCX transport is enabled. Sets the interface to be used by smx for the UCX connections. The interface should be mentioned by its name. Empty value means to use the same interface used by OpenSM. Default value: Empty.

9.2.2 Switching from UCX to Sockets

In case sockets are selected as the communication method, make sure to adjust the UFM firewall settings.

UFM Enterprise Gen 3.x includes a firewall that blocks the TCP port used by **sharp_am** by default, preventing SHARP clients from connecting when TCP is used.
To allow this communication, open the required port by running:

```
ufw allow 6126/tcp
```

9.2.3 Management Host Network Interfaces High Availability

In case the management host has multiple network interfaces, SHARP_am can operate in HA mode, automatically handling network interface failures and switching to an active interface without interrupting any activity.

HA support for the IB transport is handled by SHARP_am itself, while HA for Ethernet transport is handled by ip-bonding.

> ⚠ In the event of network failure while a new job is being established, the operation will fail. However, upcoming job requests will not be affected, and on-going jobs will continue to operate as usual.

9.2.3.1 HA Configuration

1. ib_port_guid should be set to 0 (as its default), indicating that SHARP_am should choose which port to use and which not to use.
2. allow_remote_sm - should be set to False (as its default). HA of the IB ports can operate only when SHARP_am resides on the same machines with OpenSM.
3. In case smx ucx is enabled, smx_ucx_interface should be empty (as its default), indicating that SHARP_am should choose which interface to use and which not to use.
4. In case that smx socket is enabled, ip-bonding should be configured on the management host and smx_sock_interface should be set to the bond interface.

9.3 Appendix C: HCA-to-TOR Benchmark Tool

SHARP can measure the bandwidth of the connections between a host HCA and its directly connected switch.

This bandwidth test is useful for link validation and for identifying faults in individual cables.

In contrast to typical bandwidth testing tools that assess traffic between two hosts (from one host to a switch and then to another host), SHARP benchmarks a single connection between a host and a switch, making it easier to identify any faulty connections.

9.3.1 Running the Benchmark Tool

The benchmark tool is provided in the SHARP installation folder at bin/sharp_coll_test.

It is also included with the SHARP library in both DOCA-HOST and HPC-X. You can find it in the SHARP bin directory at: $HPCX_SHARP_DIR/bin/sharp_coll_test.

To access the application help:

```
$HPCX_SHARP_DIR/bin/sharp_coll_test -h
Usage:   sharp_coll_test[_mpi]  [OPTIONS]
Options:
-h, --help              Show this help message and exit
-d, --ib-dev            Use IB device <dev:port> (default first device found)
-j, --jobid             Explicit Job ID
-i, --iters             Number of iterations to run perf benchmark
-x, --skips             Number of warmup iterations to run perf benchmark
-M, --mem_type          Memory type(host,cuda,null) used in the communication buffers format: <src memtype>:<recv
memtype>
-s, --size              Set the minimum and/or the maximum message size. format:[MIN:]MAX Default:<4:32M>
-f, --stepfactor        increment factor> multiplication factor between sizes. Default : 2
```

Note that the **mem_type** parameter allows you to run tests that generate data from either the host CPU, the GPU, or data created directly by the HCA, facilitating a pure network check.

When vheckdata directly from the HCA, the message size must be at least 16K.

9.3.1.1 Example Test Run Using the Host CPU

```
$HPCX_SHARP_DIR/bin/sharp_coll_test -M host -d mlx5_0  -s 64M:64M

HCA < -- > TOR Bandwidth test. mem_type: HOST:HOST
   #size(bytes)     Avg lat(us)     Min lat(us)     Max lat(us)     Avg BW(Gb/s)      iters
       67108864         1510.23         1466.12         1570.44           355.49        100
```

9.3.1.2 Example Test Run Using the Host GPU

```
$ $HPCX_SHARP_DIR/bin/sharp_coll_test -M cuda -d mlx5_0  -s 64M:64M

HCA < -- > TOR Bandwidth test. mem_type: CUDA:CUDA
   #size(bytes)     Avg lat(us)     Min lat(us)     Max lat(us)    Avg BW(Gb/s)      iters
       67108864         1489.44         1431.78         1533.91          360.45        100
```

9.3.1.3 Example Test Run Sending Data Directly from the HCA:

```
$ $HPCX_SHARP_DIR/bin/sharp_coll_test -M null -d mlx5_0  -s 64M:64M

HCA < -- > TOR Bandwidth test. mem_type: HCA:HCA
   #size(bytes)     Avg lat(us)     Min lat(us)     Max lat(us)    Avg BW(Gb/s)      iters
       67108864         1409.41         1392.64         1432.58          380.92        100
```

9.3.2 Operating in a PKEY-Based System

In systems configured with partition keys (PKEYs), SHARP should be set to reservation mode. This mode restricts SHARP jobs to compute nodes included within specific reservations. The benchmark tool, therefore, can only be executed on nodes that are both part of a reservation and associated with the appropriate PKEY in their reservation data.

9.3.3 Limitations and Recommendations

The sharp_coll_test tool allows for multiple simultaneous instances, enabling extensive parallel testing. However, for optimal performance and to prevent excessive load on the sharp_am, it's best to stagger test starts to avoid large surges in requests.

For environments using TCP/IP communication between clients and sharp_am, it is recommended to limit the number of sharp_coll_test processes starting simultaneously to approximately 240 (across 40 compute nodes, each with 8 HCAs). For systems using UCX communication, further configuration adjustments in the sharp.cfg file are advised to restrict this limit to around 80 tests (10 compute nodes, each with 8 HCAs).

To enable this limit under UCX, add the following line to the sharp.cfg file (the position within the file does not matter) and restart sharp_am:

```
enable_async_send FALSE
```

9.4 Appendix D: Stochastic Rounding Support

SHARP supports several mathematical rounding methods for the collective operations.

The user's application can select its desired rounding method via an environment variable. In case the selected rounding method cannot be supplied, the SHARP job request will be rejected.

The default rounding method is "Round to Nearest Even". The user can select a different method by setting the environment variable **SHARP_ROUNDING_METHOD** with one of the following options:

Parameter Value	Description
0	Round to Nearest Even. This is the default in case no method is mentioned
1	Round to Negative Infinity
2	Round to Zero
3	Round to Positive Infinity
4	Use LFSR Stochastic Rounding

When using LFSR Stochastic Rounding, a seed can also be provided by the user, and in case no seed is provided, SHARP will randomize a seed.

9.5 Appendix E: NVIDIA SHARP Cleanup

This feature enables cleaning up all SHARP-related definitions when it is no longer desired to operate with it. The cleanup helps in leveraging the full potential of the switch capabilities without allocating resources for SHARP.

Furthermore, when an error takes place due to stuck jobs, uncleaned memory, or other scenarios, a cleanup should help in solving the error without the need for a switch reboot.

To perform a cleanup, follow the steps below.

1. Stop sharp_am or make sure sharp_am is not running.
2. Verify that there are no active SHARP jobs running. In case there are, be aware that cleaning SHARP resources will terminate these jobs, so either wait for them to finish or stop them gracefully.
3. Run sharp_am with the following parameters: **`sharp_am --clean_and_exit TRUE`**
   sharp_am will clean all ANs mentioned in the smdb file and exit.
4. To verify success, look into the sharp_am log file for a message in the following format:
   `"Sent clean command to <> ANs, successes: <>, fails: <>"`

9.5.1 Running the Cleanup in UFM

When using UFM, it is recommended to use the default configuration as set in UFM, mainly to have the log output in the same path as regular sharp_am log files.

To run the clean command within UFM, please run the following command:

```
/opt/ufm/sharp2/bin/sharp_am -O /opt/ufm/files/conf/sharp/sharp_am.cfg --clean_and_exit TRUE
```

9.6 Appendix F: Running SHARP_am from HPC-X

SHARP Aggregation Manager (sharp_am) uses a configuration file from the default location */etc/sharp/sharp_am.cfg*.

If no such file exists, sharp_am will use the factory defaults.

sharp_am can also be executed using the parameter -O that provides the location of the config file:

```
$ $HPCX_SHARP_DIR/bin/sharp_am -O <desired config file path>
```

If the file does not exist, it can be created using the following command:

```
$ $HPCX_SHARP_DIR/bin/sharp_am -c /etc/sharp/sharp_am.cfg
```

The above command creates a config file with the factory default settings. Make sure the directory exists before running the command.

In order to modify the configuration settings, edit the file and change the parameter values accordingly. Some parameters require a restart of sharp_am in order to take effect, while others only require only notifying sharp_am that a change in the config file has taken place.

In the config file, every parameter has the following comment:

 # Parameter supports update during runtime: yes/no

If one of the modified parameters does not support update during runtime, then sharp_am restart is required. If not, it is sufficient to signal sharp_am with sighup (kill -1 <pid>).

9.7 Appendix G: Disabling SHARP on Specific Network Devices in OpenSM

Disabling SHARP on a specific network switch device can be performed using OpenSM device configuration file by performing the following:

1. Define a port group with the specified network device in the port groups file. The group is specified by the **pgrp_policy_file** parameter in the OpenSM configuration file.
   For example:

   ```
   port-group
   name: NON_SHARP_SWITCHES
   port-guid: 0x0002c90000000001
   end-port-group
   ```

2. Configure OpenSM to disable SHARP on the devices of the specified port groups in device configuration file specified by the **device_configuration_file** parameter in OpenSM configuration file.

   ```
   port-conf
       port-group-name: NON_SHARP_SWITCHES
       sharp-enabled: 0
   end-port-conf
   ```

3. Reload OpenSM.

10 Deployment Guide Revision History

Revision	Date	Description
3.15.0	June 2026	Removed **Using NVIDIA SHARP from DOCA-Host** appendix.
3.14.0	February 23, 2026	• Updated SHARP in Public Cloud
3.13.0	November 05, 2025	• Added Automatic Synchronization of SHARP Reservations and PKeys • Updated NVIDIA SHARP Installation • Updated Appendix B: SHARP_am Network Interfaces
3.12.0	August 05, 2025	Reorganized document structure for clarity and improved readability.

11 Release Notes Revision History

- Release Notes Change History
- Bug Fixes History

11.1 Release Notes Change History

11.1.1 Changes and New Features History

Feature/Change	Description
Rev 3.14.0	
Stochastic Rounding	Added support for stochastic rounding method - see Parameter Changes below.
Reformed SHARP Public Cloud Support	Reformed SHARP support in public cloud to simplify the PKey-to-reservation mapping and eliminate the now-obsolete SHARP Reservation REST API. For further details, see SHARP in Public Cloud.
SHARP in DOCA-Host	NVIDIA SHARP is no longer provided as part of the DOCA-Host package.
Rev 3.13.0	
Automatic Synchronization of SHARP Reservations and PKeys	Added support for automatic synchronization between the PKey database and SHARP reservations (create, modify, and delete) in multi-tenant clusters. When this mode is enabled, there is no need to invoke the SHARP reservation REST API manually. For further information, please see Automatic Synchronization of SHARP Reservations and PKeys section.
Default Client-Server Communication	New installations of SHARP_am now use UCX as the default communication method with libsharp instead of sockets. During upgrades, SHARP_am retains the previously configured communication method. If the system was previously using sockets, customers are advised to switch to UCX for improved reliability.
Default Configuration in UFM	**sharp_enable** is now set to **true** by default, eliminating the need to update the settings in the configuration file.
Bug Fixes	See Bug Fixes.
Rev 3.12.0	
Improved Handling of MAD Errors	Enhanced SHARP_am's response to MAD errors, where instead of marking a switch as entirely unusable, it now deprioritizes the switch while keeping it eligible for job selection when alternatives are limited. Cleanup still occurs when possible, reducing disruption and improving resilience.
Bug Fixes	See Bug Fixes.
Rev 3.11.0	
Q3200-RA NVIDIA Quantum-3 Switch Systems	SHARP XDR now supports an extended range of switch platforms, adding compatibility with Q3200-RA NVIDIA Quantum-3 Switch Systems alongside existing support for Q3400-RA NVIDIA Quantum-3 Switch Systems.
Bug Fixes	See Bug Fixes.

Rev 3.10.3	
XDR Support	SHARP now supports topologies utilizing ConnectX-8 network interfaces and BlackMamba Quantum-3 switches.
Switch In-Service Software Upgrade (ISSU) Support	SHARP now supports the In-Service Software Upgrade (ISSU) process, allowing switch firmware upgrades without requiring a restart. This ensures that upgrades can be performed without impacting active SHARP jobs.
Expanded REST API for SHARP Allocation	The SHARP Allocation REST API now includes the following two new commands, enabling the addition or removal of host GUIDs without needing to specify the full list of existing host GUIDs: • add_guids • remove_guids
Non-Root Operation in UFM	The `sharp_am` service can now run with ufmapp privileges instead of requiring superuser access when operating within UFM.
Enhanced Logging for SHARP Jobs	Libsharp can now print statistics on aggregated data at the end of a SHARP job.
Rev 3.10.2	
Adapter Cards	Added support for ConnectX-8 SuperNIC.
Rev 3.9.0	
Enhanced Network Performance Benchmarking with SHARP	Added support for sharp_coll_test tool, which enables users to benchmark the network performance between compute node HCAs and TOR switches effectively.
SHARP Client Errors Display in SHARP AM	MAD errors from SHARP clients are now displayed in SHARP AM logs to facilitate monitoring of such events.
Bug Fixes	See Bug Fixes.
Rev 3.8.0	
DragonFly (DFP+) Topologies with Dynamic Trees Support	Added support for DFP+ topologies while `sharp_am` operates in dynamic trees mode.
SHARP Reservation Resource Limit	Modified the logic of SHARP resource limit per reservation by adding a new parameter to control the number of jobs per HCA (see reservation_max_jobs_per_hca below). For further information, please see SHARP Application Awareness section.
Bug Fixes	See Bug Fixes.
Rev 3.7.0	
Expanded SHARP Jobs Capacity	Previously restricted to a maximum of 1023 simultaneous operations, the capacity for SHARP jobs running concurrently has now been enhanced. The new limit is determined by the size of the cluster and the available switch resources
Security QKey	SHARP now supports the activation of a security QKey on compute nodes, ensuring a heightened level of security during operation.
SHARP Telemetry Reports	Added support for telemetry reports, delivering valuable insights at specified intervals.

Parameter Changes	See Parameters Change History below.
Bug Fixes	See Bug Fixes.
Rev 3.6.0	
Parameter Changes	smx_keepalive_refresh_interval smx_keepalive_min_time_before_connection_refresh smx_keepalive_min_percentage_of_connections_to_refresh_at_iteration
Bug Fixes	See Bug Fixes section.
Rev 3.5.1	
New SHARP capability	Added configuration parameters that control the desired behavior with regard to reservation scale-in and override of one reservation by another.
Parameter Changes	load_reservation_files reservation_force_guid_assignment reservation_stop_jobs_upon_scale_in
Bug Fixes	See Bug Fixes section.
Rev 3.5.0	
Parameter Changes	Added support for controlling the **sharp_am log** messages verbosity level per the desired log category.
Bug Fixes	See Bug Fixes section.
Rev 3.4.0	
Parameter Changes	dynamic_tree_allocation sharp_am A boolean parameter, indicates whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Update**: Default value is now True
Bug Fixes	See Bug Fixes section.
Rev 3.3.0	
Syslog Capabilities	Added support for a new syslog capability to libsharp. Syslog verbosity level can now be controlled using the SHARP_SYSLOG_VERBOSITY environment variable.
Dynamic Trees Allocation Algorithms	Added support for selecting one of two algorithms that determine how trees should be created for each SHARP job. One algorithm is optimized for SuperPOD fabrics, while the other is optimized for Quasi Fat Trees (QFTs). For further information, please see Dynamic Trees Allocation Algorithms section.
REST API Jobs Query	Added support for retrieving the status of the current active SHARP jobs along with the structure of the trees assigned to them. Note that this information is retrieved via REST-API and requires the use of UFM.
Unhealthy Ports	Added support in OpenSM to inform SHARP of dangling or unhealthy links in order to avoid their use in SHARP jobs.
Bug Fixes	See Bug Fixes section.
Rev 3.2.0	
High Availability in sharp_am Network Interfaces	sharp_am leverages multiple network interfaces of the management host to provide high availability in case of a network interface failure. For further information, please see Appendix B: SHARP_am Network Interfaces.

Reliable Multicast	Added support for SHARP to leverage reliable multicast option with NVIDIA Quantum-2.
SM Data	Removed support for reading sm data by a client application. The API functions sharp_request_sm_data, sharp_get_sm_data_buf_len, and sharp_get_sm_data have been removed and can no longer be used. In addition, the configuration parameter ftree_ca_order_file is ignored in sharp_am.
Bug Fixes	See Bug Fixes section.
Rev 3.1.1 LTS	
SHARP Cleanup	Added the ability to clean up all SHARP-related definitions either to spare resources or to contribute to the recovery from an error.
General	Updated MLNX_OFED and firmware versions in General Information section.
Bug Fixes	See Bug Fixes section.
Rev 3.1.0	
Aggregation Manager (AM)	Added support for dynamic creation of trees instead of static allocation when SHARP is initialized.
Rev 3.0.1	
Bug Fixes	See Bug Fixes section.
Rev 3.0.0	
General	Added support for executing multiple jobs that aggregate data through the same set of switches, while each job utilizes a different set of links.
	SHARP logic is now application-aware with UFM capabilities. SHARP jobs can be assigned an App-ID, which can be used as a reference to the customer application performing these jobs. For further information, please refer to UFM SLURM Integration Appendix in UFM UM.
	Added the option to limit the SHARP resources that applications are allowed to consume. For further information, please refer to UFM SLURM Integration Appendix in UFM UM.
AM	Modified the default resources provided to LLT & SAT jobs. This enables operation of a larger amount of SAT jobs in parallel to few LLT jobs (please see the first three entries in the table below).
libsharp	SHARP jobs are now executed in exclusive lock mode by default (please see SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE in the table below).
Rev 2.7.0	
Switches	Added support for NVIDIA Quantum-2 switches with NDR speed
Adapter Cards	Added support for NVIDIA ConnectX-7 adapter card with 400 Gb/s speed
SHARPD	sharpd daemon process has been removed. sharpd-related activity is now performed from the user application process
AM	Upon restart of AM, it no longer needs to wait for all concurrent jobs to finish before being able to accept new jobs
	Added a mechanism that periodically checks for errors in Aggregation Trees and attempts to fix them

General	Added support for new data types BFLOAT16, INT8 and UNIT8 for performing reduction operations
Rev 2.6.1	
General	Added support for running libsharp_coll from SHARP 2.6.1 with SHARPD from SHARP 2.4.0 – 2.6.1
General	Added information about updatable configuration parameters in the configuration file and help menu
Network	Added support for keep-alive on connections to SHARPD
Network	Added support for asynchronous connections
Network	Disabled UCX listener as default in SHARP Aggregation Manager
AM	Added support for the non-default subnet prefix
AM	Added support for DF+ topologies with more than two-level islands
SHARPD	Added support for caching AM address
Rev 2.5.0	
Resource Management	Added support for exclusive lock requests for streaming aggregation jobs.
Network	Enabled connection keep-alive between SHARPD and Aggregation Manager.
Rev 2.4.3	
General	Added support for identifying Aggregation Nodes based on SMDB.
General	Improved minhop tables calculation.
General	Added a new API for querying events.
Rev 2.1.4	
sharp_am/sharpd/ libsharp_coll: Streaming Aggregation	Added support for Streaming Aggregation over ConnectX-6 adapter card and Quantum switch.
libsharp_coll: GPU Accelerator	Added support for NVIDIA GPU buffers.
sharp_am: OOB	Added support for identifying the topology type from the OpenSM SMDB file.
sharp_am: Reboot	Fixed an issue where recovery failed after reboot of all switches in the cluster.
Rev 2.0.0	
sharp_am/sharpd/ libsharp_coll	Added support for the following NVIDIA Quantum switch capabilities: • Performing data operations on new data types (unsigned short, short, and short floating point data types) • 1K OST payload
sharp_am/ sharpd: Resource Management	Added support for enabling and disabling reproducibility on the job level.
sharp_am/ sharpd: Subnet Management	Added support for controlling the SA key for SA operations.
libsharp_coll: GPU Direct	Added support for CUDA GPUDirect and GPUDirect RDMA.

Rev 1.8.1	
Aggregation Manager (sharp_am): Resiliency	Added support for waiting for jobs to end prior to performing fabric reinitialization on AM startup.
Mellanox SHARP Daemon (sharpd): Out-of-Box Improvements	Socket-based is now activated by default when installed from RPM/MLNX_OFED.

11.1.2 Parameters Change History

Parameter	Component	Description
Rev 3.14.0		
rounding_method	**sharp_am**	**New Parameter:** Defines the rounding method that will be used by all SHARP jobs. Valid values: 0 - Round to Nearest Even (RNE) 1 - Round to Negative Infinity 2 - Round to Zero 3 - Round to Positive Infinity 4 - LFSR Stochastic Rounding. Default: The default value is 0 – RNE.
reservation_auto_by_pkeys	**sharp_am**	**Deprecated parameter.** This parameter was used to define whether SHARP reservations should be created automatically from the definitions of the pkeys in the system. It is deprecated since now SHARP now treats the pkeys automatically without any need to perform additional SHARP reservation REST APIs.
Rev 3.13.0		
smx_protocol	sharp_am	Defines the default protocol to be used when communicating with libsharp. Value 1 - UCX. Value 2 - Sockets. **Default**: The default value is modified in this release from Sockets to UCX.
reservation_auto_by_pkeys	sharp_am	**New parameter:** A boolean parameter. Defines whether SHARP reservations should be created automatically from the definitions of the pkeys in the system. **Default**: False.
SHARP_COLL_JOB_CREATE_TIMEOUT	libsharp	Defines the timeout for sharp job create (in milliseconds). **Default**: The default value is modified in this release from 10,000 milliseconds to 30,000 milliseconds.
Rev 3.12.0		

dynamic_tree_allocation	sharp_am	**Description:** A boolean parameter, tells whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Change**: This parameter is now obsolete, with dynamic allocation being the only possible mode.
Rev 3.11.0		
dynamic_tree_allocation	sharp_am	**Description:** A boolean parameter, tells whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Change**: This parameter is becoming obsolete, in the future it will be removed and dynamic allocation will be the only possible mode.
Rev 3.10.3		
smx_enabled_protocols	sharp_am	**Parameter Removed.** This parameter defined the protocols enabled by SMX (Sockets, UCX, and Unix Domain Socket). Unix Domain Socket is now always enabled, and Sockets and UCX no longer need to be enabled simultaneously—only one is required. The choice between Sockets and UCX is now controlled by the existing smx_protocol configuration parameter.
SHARP_SMX_SOCK_ADDR_FAMILY	libsharp	**New parameter:** A string, defines whether libsharp should communicate via IPv4, IPv6 or make an automatic decision. This environment variable is relevant only when sockets are used for SMX communication. **Valid values**: ipv4, ipv6, auto **Default**: auto.
SHARP_COLL_STATS_DUMP_MODE	libsharp	**New parameter:** An enum value, defines whether libsharp should print statistics at the end of a job. Valid values: 0 - Do not print stats. 1- Print stats of rank 0. 2 - Print stats of all the processes. **Note**: The stats are printed at the same place, but will include information about all the participating processes). **Default**: 0 - Do not print stats.
SHARP_COLL_STATS_FILE	libsharp	**New parameter:** A string, telling the destination of the printed stats. **Valid values**: • stdout - Print to standard output. • stderr - Print to standard error. • file:<filename> - Save to a file by the <filename>. Escape characters can be used in the file name: %h: host, %p: pid, %t: time, %u: user, %e: exe. **Default**: stdout - Print to standard output.
Rev 3.9.0		
log_verbosity	sharp_am	Sets the sharp_am log verbosity. The default value is modified from Warning level to Info level.
ib_sa_key	sharp_am	Parameter is no longer supported. Used to control the SA Key that sharp_am was using. This information was provided automatically from the SM.
Rev 3.8.0		

reservation_max_jobs_per_hca	sharp_am	**New parameter:** A numeric parameter. Tells the maximum number of allowed jobs that can use the same HCA. A value of 0 means no limit. Valid range: 0-511. Applies only while operating in reservation mode. **Default**: 1 job per HCA.
dynamic_tree_algorithm	sharp_am	Sets which algorithm should be used by the dynamic tree mechanism. Modified value 1 to include support for DragonFly topologies. Current values: 0 - Regular FatTree oriented algorithm 1 - Quasi Fat Tree or DragonFly oriented algorithm
Rev 3.7.0		
rdma_sr_enable	sharp_am	**New parameter:** A boolean parameter. Tells whether **sharp_am** should provide its own service record via **rdmacm** service, enabling **libsharp** to find **sharp_am** even when a security QKey is enabled. **Default**: True.
telemetry_interval	sharp_am	**New parameter:** A decimal parameter. Tells the interval in seconds between **sharp_am** telemetry updates. A value of 0 means no telemetry reports. Valid range of values: 0, 10-3600 **Default**: 60 seconds.
telemetry_file_path	sharp_am	**New parameter:** A string parameter. Tells the full path of the **sharp_am** telemetry file output. An empty path or (null) means no telemetry reports. **Default in UFM**: **/opt/ufm/log/sharp_am_telemetry.dump** **Default in non UFM systems**: (null)
smx_sock_addr_family	sharp_am	Determines which address family will be used by SMX's sockets. New option is added, the current possible options are: **auto, ipv4, ipv6** . The new " **auto** " option means that both IPv4 and IPv6 can be used if applicable, and if only one of them is configured on the management host, then the configured address will be used. **Default**: auto.
SHARP_SMX_SOCK_ADDR_FAMILY	libsharp	**Parameter Removed.** This environment variable controlled the socket address family that libsharp used (IPv4/IPv6). The parameter is removed, since now the selection is automatic, according to the **sharp_am** supported address family.
SHARP_USE_USER_QKEY	libsharp	**New parameter:** A boolean parameter. Tells whether libsharp should use user QKey for MAD QPs. In case that a compute node is configured with security qkey enabled, then sharp should use a user Qkey and this environment variable should be set to **true** . **Default**: False
SHARP_SR_QUERY_SOURCE	libsharp	**New parameter:** Defines the source that should be used in order to fetch the sharp_am service record. Possible values: 0 - Fetch only from the SA (opensm), this was the only supported option before sharp version 3.7.0. 1 - Fetch only from Sharp_am itself (requires that **sharp_am** is configured with **rdma_sr_enable** = **true**). 2 - Try both options, try first from SA (OpenSM) and if not successful, try from Sharp_am. **Default**: 2 - Try both options.

Rev 3.5.0		
log_categories_file	**Sharp_am**	Added support for a new string parameter which enables indicating the log categories file path. The value "(NULL)" indicates that the log categories file does not exist. **Default**: In UFM, the default path is: **/opt/ ufm/files/conf/fabric_log_categories.cfg**
Rev 3.3.0		
dynamic_tree_algorithm	sharp_am	**New parameter**: Sets which algorithm should be used by the dynamic tree mechanism. This parameter is ignored when dynamic_tree_allocation is false. Possible values: 0 - SuperPOD oriented algorithm 1 - Quasi Fat Tree oriented algorithm **Default**: 0 – SuperPOD oriented algorithm
app_resources_default_limit	sharp_am	Sets the default max number of trees allowed to be used in parallel by a single app. Modified the possible range of values where the value of –1 means no resource limit, and 0 means no resources by default. **Default**: -1 – No resource limit
max_quota	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated. It is ignored and should not be used.
default_quota	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated. It is ignored and should not be used.
SHARP_SYSLOG_VERBOSITY	libsharp	**New parameter**: Sets the libsharp syslog verbosity level. Possible values: 0 – Disable syslog 1 – Errors log level 2 – Warnings log level 3 – Info log level **Default**: 1 – Errors log level
SHARP_GROUP_JOIN_MAD_TIMEOUT	libsharp	Sets the timeout till a retry for GroupJoin MAD, in milliseconds. Modified the default value. **Default**: 3000 milliseconds
SHARP_GROUP_JOIN_MAD_RETRIES	libsharp	Sets the number of retries for GroupJoin MAD. Modified the default value. **Default**: 5 retries
SHARP_QP_CONFIRM_MAD_TIMEOUT	libsharp	Sets the timeout till a retry for QP Allocation confirmation MAD, in milliseconds. Modified the default value. **Default**: 2000 milliseconds
Rev 3.2.0		
ignore_host_guids_file	sharp_am	**New parameter**: File with a list of Host GUIDs to be ignored for SHARP trees. **Default**: Null.
ignore_sm_guids	sharp_am	**New parameter**: A boolean parameter, telling whether SM GUIDs need to be ignored in SHARP trees parsed from SMDB file. **Default**: True.

ftree_ca_order_file	sharp_am	**Deprecated parameter**: This parameter is now marked as deprecated, it is ignored and should not be used.
enable_sat	sharp_am	**Deprecated parameter**: This parameter controlled whether SHARP should allow SAT jobs. The parameter is now marked as deprecatel. it is ignored and should not be used. SAT is always supported.
SHARP_COLL_SERIALIZE_MADS	libsharp	**New parameter**: Serialize sharp MADs in tree connect and group join operations, it is recommended to set this flag to true when running mpirun with multiple groups. **Default**: False.
SHARP_COLL_JOB_REQUEST_RMC	libsharp	**New parameter**: If set to True, require that any allocated SHARP trees will support the Reliable Multicast feature. **Default**: False.
SHARP_COLL_FORCE_BCAST_AS_ALLREDUCE	libsharp	**New parameter**: Force Bcast(rmc) as Allreduce operation **Default**: False.
Rev 3.1.1 LTS		
clean_and_exit	sharp_am	**New parameter:** A boolean parameter. When set to TRUE, sharp_am does not operate normally, but instead cleans SHARP resources from all switches and exits. **Default**: False - Operate normally.
Rev 3.1.0		
dynamic_tree_allocation	sharp_am	**New parameter:** A boolean parameter, tells whether trees should be allocated dynamically for each SHARP job or have trees allocated during sharp_am initialization. **Default**: False
max_trees_to_build	sharp_am	**Update**: In case dynamic_tree_allocation is set to True, this parameter will have no effect on the number of trees allocated; sharp_am would determine that value based on the amount of possible trees the switches can have. However, in the dynamic trees mode, this parameter affects the number of skeleton trees that sharp_am will use. It is recommended that the minimal value be the same as the number of root switches in the fabric. In case dynamic_tree_allocation is set to False, this parameter can be used to fulfil its purpose. **Default**:
SHARP_COLL_IB_TIMEOUT	libsharp	**New parameter:** Transport timeout on SHARP QP **Default**: 18
SHARP_COLL_IB_RETRY_COUNT	libsharp	**New parameter:** Transport retries on SHARP QP **Default**: 7
SHARP_COLL_IB_RNR_TIMER	libsharp	**New parameter:** RNR timeout on SHARP QP **Default**: 12

SHARP_COLL_IB_RNR_RETRY	libsharp	**New parameter:** RNR retries on SHARP QP **Default**: 7
SHARP_COLL_IB_SL	libsharp	**New parameter:** SL **Default**: 0
SHARP_COLL_ENABLE_MCAST_TARGET	libsharp	**Update**: Modified the default value from True to False. **Default**: False
Rev 3.0.0		
per_prio_default_quota	sharp_am	**Update**: This parameter controls only the default percentage provided to LLT jobs. Its default value is modified from 3 to 20
per_prio_default_sat_quota	sharp_am	**New parameter**: Default percentage of quota (OSTs, Buffers and Groups) per aggregation node per tree, to be requested for a single SAT job by its priority. If no explicit quota request is submitted, this parameter will set the quota percentage to be used. **Format**: prio_0_quota, [prio_1_quota, ..., prio_9_quota] Note that if only one value is set, it will be applied to all priorities. **Default**: 3
sat_jobs_default_absolute_osts	sharp_am	**New parameter**: Default number of OSTs to be allocated for SAT jobs per aggregation node per tree. Zero value means that no absolute value should be used, and the default percentage value is used instead. Note that the number of OSTs also affects the number of groups. **Default**: 0
app_resources_default_limit	sharp_am	**New parameter**: A numerical parameter, applicable only when reservation_mode is set to true. Sets the default max number of trees allowed to be used in parallel by a single app. This default value can be overridden per app upon reservation request. A value of 0 means no allowed resources, which means an app cannot execute any sharp job. **Default**: 1
force_app_id_match	sharp_am	**New parameter**: A boolean parameter, applicable only when reservation_mode is set to true. When set to true, an application ID must be provided upon job request, and it must match the application ID provided upon reservation request. Otherwise, the job will be denied. **Default**: False
SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE	libsharp	**Update**: Changed default value from 0 (no exclusive lock) to 2 (force exclusive lock)
Rev 2.7.0		
recovery_retry_interval	sharp_am	**New parameter:** A timeout in seconds for trees recovery retries. A value of 0 means do not try to recover trees. **Default**: 300
enable_seamless_restart	sharp_am	**New parameter:** A boolean flag. If enabled, AM tries to recover state from last AM run and continue the operation of the current jobs. **Default**: True

seamless_restart_trees_file	sharp_am	**New parameter:** Set the SHARP trees file used in Seamless restart. Need to mention only the file name, full path is constructed using 'dump_dir'. **Default**: sharp_am_trees_structure.dump
seamless_restart_max_retries	sharp_am	**New parameter:** Set the number of consecutive retries of seamless restart. If seamless restart fails more times in a row, it will be disabled in the next run. **Default**: 3
max_tree_radix	sharp_am	**Update**: Change default to 252
Ib_sat_max_mtu	sharp_am	**Update**: Change default to 5, to support MAD value that represents 4K MTU.
per_prio_default_quota	sharp_am	**Update**: Changed default to 3 instead of 20, enabling more SAT jobs to take place in parallel on each switch.
Rev 2.6.1		
dump_dir	sharp_am	**Update**: Changed default to /var/log
smx_enabled_protocols	sharp_am	**Update**: Changed default from 7 to 6 (disable UCX by default)
ib_mad_timeout	sharp_am	**Update**: Change deault from 200 to 500
dump_dir	sharp_am	**Update**: Change default to /var/log
sr_mad_timeout	sharpd	**New parameter:** Control timeout for ServiceRecord queries **Default**: 10000 millieconds
sr_mad_retries	sharpd	**New parameter:** Control number of retries for ServiceRecord queries **Default**: 3 retires
Rev 2.5.0		
smx_keepalive_interval	sharp_am/ sharpd	**New parameter:** Keep alive interval in seconds 0 to disable keep alive. **Default:** 60 seconds
smx_incoming_conn_keepalive_interval	sharp_am	**New parameter:** Keep alive interval for incoming connections **0 to disable** **Default: 300 seconds**
enable_exclusive_lock	sharp_am	**New parameter:** Enable/Disable exclusive lock feature. **Default:** True
enable_topology_api	sharp_am	**New parameter:** Enable/Disable Toplogy API feature **Default:** True
max_trees_to_build	sharp_am	**New parameter**: Control number of trees for AM to build **Default:** 126

Rev 2.4.3		
ib_max_mads_on_wire	sharp_am	**Modified behavior**: Changed default from 100 to 4096
ib_qpc_local_ack_timeout	sharp_am	**Modified behavior**: Changed default from 0x1F to 0x12
ib_sat_qpc_local_ack_timeout	sharp_am	**Modified behavior**: Changed default from 0x1F to 0x12
ib_qpc_timeout_retry_limit	sharp_am	**Modified behavior**: Changed default from 7 to 6
ib_sat_qpc_timeout_retry_limit	sharp_am	**Modified behavior**: Changed default from 7 to 6
Rev 2.0.0		
control_path_version	sharp_am	**New parameter** **Default**
max_compute_ports_per_agg_node	sharp_am	**Modified behavior**: When set to 0, AN radix is set to maximal radix value. **Default**: 0
default_reproducibility	sharp_am	**New parameter**: Control default reproducibility mode for jobs. **Default**: TURE
ib_sa_key	sharp_am	**New parameter**: Control SA key for SA operations. **Default**: 0x1
coll_job_quota_max_payload_per_ost	sharp_job_quota	**Modified behavior**: Change default value to 1024.
SHARP_COLL_MAX_PAYLOAD_SIZE	Libsharp_coll	**Removed**
SHARP_COLL_NUM_SHARP_COLL_REQ	Libsharp_coll	**Removed**

SHARP_COLL_ENABLE_REPRODUCIBLE_MODE	Libsharp_coll	**New parameter**: Control job reproducibility mode: 0 – Use default. 1 – No reproducibility. 2 – Reproducibility.
SHARP_COLL_ENABLE_CUDA	Libsharp_coll	**New parameter:** Enables CUDA GPU direct.
SHARP_COLL_ENABLE_GPU_DIRECT_RDMA	Libsharp_coll	**New parameter:** Enables GPU direct RDMA.
Rev 1.8.1		
pending_mode_timeout	sharp_am	**New parameter**: Defines AM waiting time for jobs to complete prior to fabric re-initialization upon startup.
job_info_polling_interval	sharp_am	**New parameter**: Defines job status polling interval when waiting for jobs to complete upon startup.

11.2 Bug Fixes History

The following table provides a list of bugs fixed in this SHARP version.

Internal Ref.	Issue
4578192	**Description:** Fixed an issue where the metrics for active jobs and active SAT jobs were decreased on every job clean attempt, even when the cleanup failed, resulting in incorrect metric values.
	Keywords: Telemetry; SAT jobs
	Discovered in Version: 3.9.0
	Fixed in Release: 3.13.0
4578250	**Description:** Fixed an issue where a failure to clean a job caused the metric for jobs ended due to client failure to continuously increase, resulting in incorrect values being reported in telemetry.
	Keywords: Telemetry
	Discovered in Version: 3.9.0
	Fixed in Release: 3.13.0
4507679	**Description:** Fixed an issue where SHARP_am was ignoring SharpError traps due to missing syndrome information, leading to a flood of repeated traps. SHARP_am now handles these traps correctly and suppresses redundant resends.
	Keywords: SHARP_am; traps
	Discovered in Version: 3.11.0

Internal Ref.	Issue
	Fixed in Release: 3.12.0
4507678	**Description:** Fixed an issue where SHARP jobs failed to start due to improper handling of timeouts in QP Allocation and Confirmation MADs. The updated logic adds retries with shorter timeouts and re-attempts the full QP allocation sequence, improving start-job resiliency and reducing failure risk from network timeouts.
	Keywords: libsharp; timeouts
	Discovered in Version: 3.11.0
	Fixed in Release: 3.12.0
4259313	**Description**: Fixed an issue where the SHARP component was not upgraded correctly during a DOCA-Host upgrade to version 3.0.0 from earlier versions on RedHat and SLES systems. **Note**: This fix applies to upgrades from version 3.0.0 onward. Upgrades to 3.0.0 from prior versions will still experience the issue. Workaround for affected upgrades: To ensure SHARP is updated when upgrading to version 3.0.0: • **On SLES**: run `zypper up doca-ofed sharp` • **On RedHat**: run `dnf update doca-ofed sharp`
	Keywords: DOCA-Host; upgrade
	Discovered in Release: 3.10.3
	Fixed in Release: 3.11.0
4410095	**Description:** Fixed an issue where, when a QPError trap was reported (typically due to a physical link failure), SHARP_am failed to clean up the relevant SHARP job.
	Keywords: SHARP_am; QPError trap
	Discovered in Version: 3.9.0
	Fixed in Release: 3.9.1
4381790	**Description:** Fixed an issue where SHARP traps from the switch were neither handled nor acknowledged by SHARP_am, resulting in repeated retransmission of the same trap.
	Keywords: SHARP_am; switch; traps
	Discovered in Version: 3.9.0
	Fixed in Release: 3.9.1
4068969	**Description:** Fixed a rare issue where a failed SHARP job request to configure a required switch would result in job initialization failure without complete cleanup. This led to repeated log messages about unsuccessful cleanup attempts. The fix ensures that the job is properly cleaned up, even if errors occur during initialization.
	Keywords: sharp_am
	Discovered in Version: 3.6.0
	Fixed in Release: 3.9.0
3844898	**Description:** Fixed the issue where sharp_am failed to allocate resources for new job requests due to scattered links and unmatched trees, despite a sufficient number of links available.
	Keywords: SHARP, Report No resource
	Discovered in Version: 3.5.1

Internal Ref.	Issue
	Fixed in Release: 3.8.0
3438393	**Description:** Fixed the issue where, in the following configuration mode, resource limitations were ignored and no limits were set for any application: when using dynamic trees allocation, Quasi Fat Tree (QFT)-oriented logic, and reservation_mode is enabled.
	Keywords: Dynamic trees allocation; QFT; resource limitation
	Discovered in Release: 3.3.0
	Fixed in Release: 3.8.0
3971970	**Description:** Fixed the issue where `sharp_am` incorrectly sent a Syslog message indicating it was shutting down, despite being in the startup phase and functioning correctly.
	Keywords: Syslog
	Discovered in Release: 3.5.0
	Fixed in Release: 3.8.0
3478803	**Description:** Fixed the issue where obtaining topology information (`sharp_cmd topology`) failed when executed from the management host.
	Keywords: SHARP topology API
	Discovered in Release: 3.5.0
	Fixed in Release: 3.8.0
3844898	**Description:** Fixed the issue where sharp_am failed to allocate resources for new job requests due to scattered links and unmatched trees, despite a sufficient number of links available.
	Keywords: SHARP, Report No resource
	Discovered in Release: 3.5.1
	Fixed in Release: 3.7.0
3696666	**Description**: Fixed the issue where libsharp could not communicate with sharp_am on systems that exclusively used IPv6 addresses without IPv4 addresses. Now, both libsharp and sharp_am can utilize either IPv4 or IPv6, depending on the machine configuration.
	Keywords: sharp_am, libsharp, tcp/ip, smx
	Discovered in Release: 3.5.1
	Fixed in Release: 3.7.0
3686321	**Description:** When upgrading UFM from previous versions to UFM 6.15.x, `sharp_am` persistent directory as mentioned in the configuration file directs to a path that does not exist. This leads to failure in saving reservation and job information, so in case of a restart of `sharp_am` , it won't be able to retrieve required information and return to its previous state.
	Keywords: `sharp_am` , UFM, upgrade
	Discovered in Release: 3.5.0
	Fixed in Release: 3.6.0

Internal Ref.	Issue
3724093	**Description**: Fixed the issue where libsharp, when communicating with sharp_am via UCX, automatically selects the first available IB adapter instead of the instructed adapter for the data path.
	Keywords: `libsharp` , UCX
	Discovered in Release: 3.5.1
	Fixed in Release: 3.6.0
3665349	**Description**: Fixed an issue where sharp_am failed to detect an abnormal termination of an application executing a SHARP job, which resulted in the failure to properly clean up its resources.
	Keywords: `sharp_am` , `libsharp`
	Discovered in Release: 3.6.0
	Fixed in Release: 3.6.0
3646010	**Description**: Fixed an issue in sharp_am where it failed to support virtual ports when OpenSM topology policies were employed, and sharp_am was configured to utilize only one of the sub-topologies.
	Keywords: `sharp_am` , Virtual Ports, OpenSM, Topology Policy
	Discovered in Release: 3.6.0
	Fixed in Release: 3.6.0
3609384	**Description**: Fixed issues concerning `Sharp_AM` connection creation with rank zero clients of active jobs during a restart when UCX is enabled.
	Keywords: `sharp_am` , libsharp, restart
	Discovered in Release: 3.4.0
	Fixed in Release: 3.5.0
3541153	**Description**: Fixed an issue where client application is abnormally terminated before the sharp_coll_finalize method, `sharp_am` is supposed to automatically detect and clean the job resources. However, with UCX, only one such termination is detected per cycle, leading to incomplete job cleaning. Similarly, when using NCCL and hosts with multiple GPUs/HCAs, each HCA gets its own SHARP job, which results in `sharp_am` taking several cycles to detect all the jobs that require cleaning. As a consequence, hosts operating in the previous application cannot initiate a new SHARP job until sharp_am detects and cleans all the necessary jobs.
	Keywords: `sharp_am` , NCCL, UCX
	Discovered in Release: 3.4.0
	Fixed in Release: 3.5.0
3400293	**Description:** Fixed an issue in libsharp where it failed to respond to messages from the SM while searching for Service Records, causing the SM to print timeout messages.
	Keywords: sharp_am; openSM
	Discovered in Release: 3.1.0
	Fixed in Release: 3.4.0
3479721	**Description:** Fixed the issue where sharp_am did not handle hypercube topologies well, causing it to incorrectly treat different switches as duplicates.

Internal Ref.	Issue
	Keywords: sharp_am; hypercube
	Discovered in Release: 3.3.0
	Fixed in Release: 3.4.0
3496440	**Description:** Fixed the issue in sharp_am where excessive log messages were printed for each disconnected or restarted compute host. Now, the information is printed in a consolidated manner in the form of summaries of disconnected hosts or a list of those hosts in a single log message. However, for more comprehensive details, the complete list of hosts is still available and printed at the DEBUG level.
	Keywords: sharp_am
	Discovered in Release: 3.3.0
	Fixed in Release: 3.4.0
3336788	**Description:** Fixed the issue in Firmware where MAD error responses might have been received in libsharp.
	Keywords: sharp_am; libsharp
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0 (Quantum-2 Firmware 31.2010.6064)
3343503	**Description:** Fixed the issue where sharp_am installed from MLNX_OFED used an invalid range of job IDs, resulting in occasional errors when trying to establish new SHARP jobs.
	Keywords: MLNX_OFED; sharp_am
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0
3368381	**Description:** Fixed the issue of when no sufficient amount of retries was made to resend failed libsharp GroupJoin MADs, SHARP jobs failed before they even started.
	Keywords: libsharp; MADs
	Discovered in Release: 3.0.0
	Fixed in Release: 3.3.0
3393902	**Description:** Fixed the issue where re-created virtual ports were not recognized by sharp_am, thus the correct tree was not built for them. This resulted in SAT jobs getting ibv_poll_cq failure in libsharp.
	Keywords: Virtual port; sharp_am; libsharp; SAT; ibv_poll_cq
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0
3404474	**Description:** Fixed an issue where failure of application allocation of all hosts done via /app/sharp/resources REST-API returned a successful job instead of error.
	Keywords: REST API; allocation
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0
3406186	**Description:** Fixed an issue where SHARP AM failed handling reports from OpenSM if some switch ports were down or isolated.
	Keywords: Aggregation Manager; Aggregation Node; OpenSM

Internal Ref.	Issue
	Discovered in Release: 3.2.0
	Fixed in Release: 3.3.0
3236363	**Description:** Fixed the way physical link failures between switches are handled. In the event of a link failure, a SHARP job utilizing the link has to be stopped; however, this will bear no effect on the other present or future jobs.
	Keywords: Aggregation Manager; sharp_am; Link Failure
	Discovered in Release: 3.1.0
	Fixed in Release: 3.2.0
3230585	**Description:** Fixed the issue of when operating in Dynamic trees mode, ibdiagnet may have printed warning messages about the existence of multiple distinct trees with the same tree ID.
	Keywords: Dynamic tree; ibdiagnet
	Discovered in Release: 3.1.0
	Fixed in Release: 3.2.0
3226743	**Description:** Fixed the issue of when a management host was not connected to a leaf switch, sharp_am might have printed a number of warning messages about trees that could not reach all aggregation nodes. As of SHARP v3.2.0, the active management host is automatically identified and is not treated as a potential compute host. However, please note that this does not include standby management hosts for which a warning message would still appear. These management hosts can be mentioned in a list of GUIDs to ignore via the parameter ignore_host_guids_file.
	Keywords: Aggregation Manager; sharp_am; leaf; GUID
	Discovered in Release: 3.0.1
	Fixed in Release: 3.2.0
3274564	**Description:** Fixed an issue where sharp_benchmark bash script failed to operate on all bash versions.
	Keywords: sharp_benchmark
	Discovered in Release: 3.1.1
	Fixed in Release: 3.2.0
3262936	**Description:** Fixed the issue where a crash took place during sharp_am reboot while physical links were hanging between switches in the fabric.
	Keywords: sharp_am; physical links; crash
	Discovered in Release: 3.1.0
	Fixed in Release: 3.1.1 LTS
3192770	**Description:** Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.
	Keywords: SR-IOV
	Discovered in Release: 3.0.0
	Fixed in Release: 3.1.0

Internal Ref.	Issue
3163697	**Description:** Fixed the issue of when the client application used more than 1024 file descriptors (range limit defined by FD_SETSIZE), libsharp was prevented from using any more file descriptors. Using poll() instead of select() enables using the full range of allowed file descriptors by Linux.
	Keywords: File descriptor; libsharp; HCOLL; HPC-X
	Discovered in Release: 3.0.0
	Fixed in Release: 3.1.0
3192770	**Description:** Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.
	Keywords: SR-IOV
	Discovered in Release: 3.0.0
	Fixed in Release: 3.0.1
3163697	**Description:** Fixed the issue of when the client application used more than 1024 file descriptors (range limit defined by FD_SETSIZE), libsharp was prevented from using any more file descriptors. Using poll() instead of select() enables using the full range of allowed file descriptors by Linux.
	Keywords: File descriptor; libsharp; HCOLL
	Discovered in Release: 3.0.0
	Fixed in Release: 3.0.1
2995739	**Description:** Sharp_am daemon is no longer removed when performing rpm upgrade and is overridden instead.
	Keywords: Aggregation Manager; rpm
	Discovered in Release: 2.6.1
	Fixed in Release: 2.7.0
2972970	**Description:** Fixed the issue where completion of SHARP installation using sharp_daemons_setup.sh script depended on python availability.
	Keywords: Aggregation Manager
	Discovered in Release: 2.6.1
	Fixed in Release: 2.7.0
2749073	**Description:** SHARP AM reports the rediscovery of aggregation nodes on every topology change.
	Keywords: Aggregation Manager
	Workaround: N/A
	Discovered in Release: 2.5.0
2736102	**Description:** SHARP AM and SHARPD overrides backlog files after restart when log rotation is enabled.
	Keywords: Aggregation Manager, SHARPD, log file
	Workaround: N/A
	Discovered in Release: 2.5.0

Internal Ref.	Issue
2700530	**Description:** Terminating a job process during job initialization before sending a job request to Aggregation Manager, might result in job resource leakage in the SHARP Aggregation Manager.
	Workaround: N/A
	Keywords: SHARPD, Aggregation Manager
	Discovered in Release: 2.5.0
2726821	**Description:** Terminating SHARPD while the job process is still running will result in job resource leakage in SHARP Aggregation Manager.
	Workaround: Terminate SHARPD after terminating the job processes.
	Keywords: SHARPD, Aggregation Manager
2795902	**Description:** SHARPD might allocate handlers on GPU when running with UCX.
	Keywords: SHARPD, SMX, UCX
	Workaround: N/A
	Discovered in Release: 2.5.0
	Workaround: Disable UCX
2770210	**Description:** Syslog verbosity depends on log file verbosity.
	Keywords: SHARPD, Aggregation Manager
	Discovered in Release: 2.5.0
	Workaround: None
2825519	**Description:** Aggregation Manager continue to run after SM failover.
	Keywords: Aggregation Manager
	Discovered in Release: 2.5.0
	Workaround: Stop AM daemon manually
2754175	**Description:** SHARP Aggregation Manger might allocate bad links for jobs after receiving timeouts from Aggregation Nodes.
	Workaround: Restart corresponding switch or restart SHARP Aggregation Manager.
	Keywords: Aggregation Manager
	Discovered in Release: 2.5.0
2796317	**Description:** SHARP jobs may hang when running in reservations mode (i.e. SHARP allocation is enabled), and reservation is created with limited PKEY, and configuring reservation PKEY on tree is enabled.
	Workaround: The PKEY used for creating the reservation should be "full" (the most significant bit should be on e.g. 0x805c instead of 0x5a).
	Keywords: Aggregation Manager, Reservations, PKEY, UFM
	Discovered in Release: 2.5.0

12 Legal Notices and 3rd Party Licenses

Product	Version	Legal Notices and 3rd Party Licenses
NVIDIA SHARP	3.15.0	• License • 3rd Party Notice • 3rd Party Unify Notice

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.

NVIDIA