



# **NVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP) Rev 3.10.2**

# Table of contents

Release Notes	4
General Information	4
Changes and New Features	5
Bug Fixes in this Version	6
Known Issues	6
Introduction	11
Setting up NVIDIA SHARP Environment	12
Running NVIDIA SHARP Aggregation Manager (AM) Daemons	15
Modifying NVIDIA SHARP Aggregation Manager Configuration	18
sharp_am Network Interfaces	20
sharp_am Log and Dump Files	24
Operating NVIDIA SHARP in Dynamic Trees Allocation Mode	27
SHARP Reservation	30
Operating NVIDIA SHARP with PKeys	32
Disabling SHARP on Specific Network Devices in OpenSM	34
Testing NVIDIA SHARP Setup	35
NVIDIA SHARP Collective Library	40
Using NVIDIA SHARP with Open MPI	46
Using NVIDIA SHARP with NVIDIA NCCL	49
Using NVIDIA SHARP with UCC	54

NVIDIA SHARP Cleanup	57
sharp_am Telemetry	58
HCA-to-TOR Benchmark Tool	61
Deployment Guide Revision History	65
Release Notes Revision History	68
Release Notes Change History	68
Bug Fixes History	82

## 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 3<sup>rd</sup> 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](#)

## Document Revision History

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

---

# Release Notes

Revision	Date	Description
3.10.2	December 24, 2024	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](#)

## General Information

### Packages Provided with SHARP

SHARP Software is provided with the following package:

Package	Version
DOCA-Host	2.9.0 <b>NOTE</b> : Starting this version, the host driver is part of the NVIDIA DOCA package. For further information, please see <a href="#">NVIDIA MLNX_OFED to DOCA-OFED Transition Guide</a> .
HPC-X	2.21.0
UFM (Aggregation Manager only)	6.19.5

### Prerequisites

## Note

NVIDIA SHARP requires either NVIDIA Unified Fabric Manager (UFM®), or a dedicated server running Subnet Manager. In the latter case, the onboard Subnet Manager should be disabled in managed switches.

Name	Firmware/Software Version
NVIDIA Quantum	27.2014.2084
NVIDIA Quantum-2	31.2014.2084
ConnectX-6	20.38.1900
ConnectX-6 DE	22.38.1900
ConnectX-7	28.38.1900
ConnectX-8	40.44.0208
MLNX-OS	3.12.1002
Subnet Manager	OpenSM 5.20

## Supported Operating Systems and Platforms

For complete list of supported Operating Systems and platforms, please refer to list of operation systems and platforms supported by [HPC-X](#) and [DOCA-Host](#).

# Changes and New Features

## Changes and New Features

Feature/Change	Description
Adapter Cards	Added support for ConnectX-8 SuperNIC.

## Parameter Changes

N/A

## Bug Fixes in this Version

N/A

## Known Issues

Internal Reference Number	Issues
4381790	<b>Description:</b> When the Switch FW is of version 31_2014_3000 or newer, SHARP_am fails to recognize the traps sent from the switch, resulting in repeated retransmission of the same trap, and seeing the following message repeatedly in SHARP_am log: "Received unsupported vendor specific trap".
	<b>Workaround:</b> A switch reboot will stop any active trap retransmission. To fix the issue, upgrade SHARP_am version to 3.11.0 or newer (UFM version 2.23 or newer).
	<b>Keywords:</b> SHARP_am; switch; traps
	<b>Discovered in Version:</b> 3.9.0
3340353	<b>Description:</b> 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.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> active; standby; compute host; ignore_sm_guids
	<b>Discovered in Release:</b> 3.3.0
3371820	<b>Description:</b> Congestion Control cannot be configured on the same SLs used by sharp_am.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> Congestion control; SL

Internal Reference Number	Issues
	<b>Discovered in Release:</b> 3.3.0
3305335	<p><b>Description:</b> When running mpirun with multiple groups, the following error message might be received:  <pre>[error] - AM QPAlloc confirm QP MAD response status 0x1c00</pre> This message is received due to the fact that multiple unserialized MAD requests are run in parallel.</p> <p><b>Workaround:</b> Set the SHARP_COLL_SERIALIZE_MADS environment variable to TRUE when running mpirun.</p> <p><b>Keywords:</b> mpirun; SHARP_COLL_SERIALIZE_MADS</p> <p><b>Discovered in Release:</b> 3.2.0</p>
3225401	<p><b>Description:</b> 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.</p> <p><b>Workaround:</b> N/A</p> <p><b>Keywords:</b> Aggregation Manager; sharp_am; dynamic trees</p> <p><b>Discovered in Release:</b> 3.1.0</p>
3237831	<p><b>Description:</b> 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.</p> <p><b>Workaround:</b> N/A</p> <p><b>Keywords:</b> Aggregation Manager; OpenSM</p> <p><b>Discovered in Release:</b> 3.1.0</p>
3048427	<p><b>Description:</b> 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.</p> <p><b>Workaround:</b> Restart sharp_am after changing a switch split mode definition.</p> <p><b>Keywords:</b> Aggregation Manager; split mode</p> <p><b>Discovered in Release:</b> 2.7.0</p>

Internal Reference Number	Issues
3051699	<p><b>Description:</b> 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.</p> <p><b>Workaround:</b> 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.</p> <p><b>Keywords:</b> Aggregation Manager; split port</p> <p><b>Discovered in Release:</b> 2.7.0</p>
3051924	<p><b>Description:</b> Adding or replacing non-leaf switches is currently not supported by Aggregation Manager for Dragonfly+ topologies.</p> <p><b>Workaround:</b> Restart Aggregation Manager after the Subnet Manager completes fabric reconfiguration followed by the fabric changes.</p> <p><b>Keywords:</b> Fabric extension; Aggregation Manager; AM</p> <p><b>Discovered in Release:</b> 2.7.0</p>
-	<p><b>Description:</b> On multi PKEY environment, UCX in SHARP can use only the default PKEY (PKEY at index 0).</p> <p><b>Workaround:</b> Use sockets for communication over non-default PKEY.</p> <p><b>Keywords:</b> Configuration, SMX, UCX, PKEY</p> <p><b>Discovered in Release:</b> 2.4.3</p>
1307124	<p><b>Description:</b> Begin Job requests with virtual ports might be rejected until fabric virtualization info file is parsed.</p> <p><b>Workaround:</b> Wait for AM to discover virtual ports before sending Begin Job requests.</p> <p><b>Keywords:</b> Aggregation Manager, Socket Direct, Virtual Ports</p> <p><b>Discovered in Release:</b> 1.5.3</p>
1193629	<p><b>Description:</b> Configuring sharp_am as daemon is not possible when installing from RPM into non-default location.</p> <p><b>Workaround:</b> Configure daemon manually.</p> <p><b>Keywords:</b> Configuration</p>

Internal Reference Number	Issues
	<b>Discovered in Release:</b> 1.5.3
1307108	<b>Description:</b> Discovering a new Aggregation Node (AN) found on the shortest path between two ANs might invalidate the existing path.
	<b>Workaround:</b> Restart Aggregation Manager after the Subnet Manager completes fabric reconfiguration followed by the fabric changes.
	<b>Keywords:</b> Aggregation Manager, Aggregation Node
	<b>Discovered in Release:</b> 1.5.3
-	<b>Description:</b> 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.
	<b>Workaround:</b> Use Aggregation Manager in UFM.
	<b>Keywords:</b> Aggregation Manager
-	<b>Description:</b> Aggregation manager should run on the same Host where the Master Subnet Manager (SM) is running.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> Aggregation Manager
-	<b>Description:</b> Aggregation Manager should be started after completion of fabric configuration by the Subnet Manager.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> Aggregation Manager
-	<b>Description:</b> Only Fat-Tree, Quasi-Fat-Tree, Hypercube and Dragonfly+ topologies are supported by the Aggregation Manager.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> Fabric Topology
-	<b>Description:</b> Only IB fabrics where all compute nodes are connected to NVIDIA SHARP capable switches are supported by the Aggregation Manager.
	<b>Workaround:</b> Manually configure mapping between the compute port and the Aggregation Node.

Internal Reference Number	Issues
	<b>Keywords:</b> Fabric Topology
-	<b>Description:</b> Upon changes in configuration file beyond parameters in 3.3, Aggregation Manager should be restarted to deploy new configuration.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> Configuration

---

# 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.

---

# Setting up NVIDIA SHARP Environment

NVIDIA SHARP binary distribution is available as part of HPC-X, DOCA-Host and UFM packages (among SHARP binaries, UFM includes Aggregation Manager (AM) only).

## Setup Requirements

Prior to installing and using NVIDIA SHARP, make sure the following requirements are met.

- Run Aggregation Manager using a "root user" as trusted entities.
- Make sure onboard Subnet Manager is disabled in the managed switches. (Aggregation Manager is a central entity running on a dedicated server with a master Subnet Manager. This dedicated server cannot serve as a compute node.
- Configure TCP/IP before running NVIDIA SHARP and Aggregation Manager communicate over TCP/IP.
- Run NVIDIA Switch-IB 2/NVIDIA Quantum/NVIDIA Quantum-2 switches with the supported firmware versions as specified in the [Prerequisites](#) section in the Release Notes (use `ibdiagnet` utility to check the installed firmware version on the switches).
- Enabled IPoIB interface in compute servers in order to enable using UD multicast for result distribution in SHARP.
- Make sure SHARP Aggregation Manager out-of-the-box subnets are configured with SM using the following routing engines:
  - Tree based topologies: `updn`, `ar_updn`, `ftree`, `ar_ftree`
  - DragonFly+ topology: `dfp`
  - Hypercube topologies: `dor` routing engine with `dor_hyper_cube_mode` enabled

## Using NVIDIA SHARP from HPC-X

When using HPC-X package, please refer to HPC-X User Manual for installation and configuration procedures.

This deployment guide includes examples on the environment variables `HPCX_SHARP_DIR` and `OMPI_HOME`, and assumes that HPC-X installation is in a shared folder accessible from all compute nodes.

To download the HPC-X packages, please visit [here](#).

## Using NVIDIA SHARP from DOCA-Host

When using DOCA-Host distribution, the `HPCX_SHARP_DIR` environment variable has to be set to redirect to SHARP installation directory (default location: `/opt/mellanox/sharp`), and `OMPI_HOME` environment variable to the MPI installation directory.

To download the DOCA-Host packages, please visit [here](#).

## Using NVIDIA SHARP Aggregation Manager from UFM

When using Aggregation Manager from UFM, NVIDIA SHARP support has to be enabled in UFM. For further information, refer to the UFM User Manual.

### Note

UFM package includes only SHARP Aggregation Manager. Other NVIDIA SHARP components are not available through UFM and should be installed from either HPC-X or DOCA-Host packages.

## NVIDIA Hardware Capabilities and Limitations

Device	Capabilities and limitations
NVIDIA Quantum	<ul style="list-style-type: none"><li>• Supports both SHARP low latency and streaming aggregation operations</li><li>• Supports up to 126 aggregation trees in the subnet (63 low latency trees, and 63 streaming aggregation trees)</li></ul>

Device	Capabilities and limitations
	<p><b>Note:</b> The number of SHARP streaming aggregation operations is limited to one active tree per switch</p>
NVIDIA Quantum-2	<ul style="list-style-type: none"> <li>• Supports both SHARP low latency and streaming aggregation operations</li> <li>• Supports up to 1023 aggregation trees in the subnet (511 low latency trees, and 511 streaming aggregation trees)</li> </ul> <p><b>Note:</b> Multiple SHARP streaming aggregation operations can be operated in parallel by a single Quantum-2 switch. The limit is one active tree per port</p>
ConnectX-5	Supports SHARP low latency operation only
ConnectX-6 and above	Supports both SHARP low latency and streaming aggregation operations

---

# Running NVIDIA SHARP Aggregation Manager (AM) Daemons

## Note

As of NVIDIA SHARP version 2.7.0, `sharpd` daemon no longer exists. `sharpd`-related activity is now performed from the user-application process instead.

This section describes how to install Aggregation Manager in the fabric using NVIDIA SHARP AM daemon script.

NVIDIA SHARP Aggregation Manager daemon (`sharp_am`) is executed on a dedicated server along with the Subnet Manager.

Installing Aggregation Manager as a service is required when used from the HPC-X or DOCA-Host packages.

## NVIDIA SHARP Daemons Installation Script

In order to install/remove NVIDIA SHARP AM daemons, use `sharp_daemons_setup.sh` script provided with the NVIDIA SHARP package. For example:

```
$HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh
```

```
Usage: sharp_daemons_setup.sh (-s | -r) [-p SHARP location dir] -  
d  
<sharpd | sharp_am> [-m]
```

```
-s - Setup SHARP daemon
-r - Remove SHARP daemon
-p - Path to alternative SHARP location dir
-d - Daemon name (sharp_am)
-b - Enable socket based activation of the service
```

## Registering sharp\_am as a Service on the Subnet Manager Node

1. Run the following as root:

```
# $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -s -d sharp_am
```

Daemon's log location is: /var/log/sharp\_am.log

2. Set the "run level".
3. Start sharp\_am as root.

```
# service sharp_am start
```

## Removing Daemons

*To remove sharp\_am, run the following on the AM host:*

```
# $HPCX_SHARP_DIR/sbin/sharp_daemons_setup.sh -r -d sharp_am
```

## Upgrading NVIDIA SHARP AM Daemons

Upgrading SHARP AM daemons requires their removal and re-registration as instructed in the sections above.



---

# Modifying NVIDIA SHARP Aggregation Manager Configuration

SHARP Aggregation Manager (`sharp_am`) has factory default configuration that can be modified either by command line parameters or through a configuration file.

`sharp_am` is operated either from UFM or HPC-X.

- In the case of UFM, `sharp_am` is provided with UFM default config file. For information on how to operate SHARP from UFM, please refer to "NVIDIA SHARP Integration" Appendix in the latest UFM User Manual available [here](#).
- In the case of HPC-X, please follow the instructions below.

## NVIDIA SHARP Integration with 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>).

---

# 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.

## 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 kept open by default for communication with UFM.

It is possible to specify certain interfaces and to change the communication protocol, using the following configuration parameters:

Parameter	Component	Description
<code>ib_port_guid</code>	<code>sharp_am</code>	Sets the GUID of the port to which <code>sharp_am</code> 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
<code>smx_enabled_protocols</code>	<code>sharp_am</code>	A bitmask specifying which transport layers should be enabled for <code>smx</code> communication. It is possible to provide multiple options. Bit 1 (value 1) - UCX Bit 2 (Value 2) - Sockets. Bit 3 (value 4) - Unix sockets (needed for UFM). Default value: 6, which means Sockets & Unix sockets.
<code>smx_protocol</code>	<code>sharp_am</code>	Defines the default protocol that will be used when communicating with <code>libsharp</code> . Value 1 - UCX. Value 2 - Sockets. Default value: 2, which means sockets.
<code>smx_sock_interface</code>	<code>sharp_am</code>	Relevant only in case that <code>smx</code> socket transport is enabled. Sets the interface to be used by <code>smx</code> 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 <code>sharp_am</code> is operating inside UFM, this parameter is automatically set by UFM according to its internal logic and the <code>am_interface</code> parameter in the <code>gv.cfg</code> file. Default value: Empty.
<code>smx_sock_port</code>	<code>sharp_am</code>	Relevant only in case that <code>smx</code> socket transport is enabled. IP port number to be used for the socket listener. Default value: 6126
<code>smx_sock_addr_family</code>	<code>sharp_am</code>	Relevant only in case that <code>smx</code> socket transport is enabled.

Parameter	Component	Description
		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
<code>smx_ucx_interface</code>	<code>sharp_am</code>	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.

## 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.

### Note

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.

## 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.

## UFM Appliance Firewall Settings

UFM Appliance Gen 3.x uses firewall that is configured to block the TCP port used by sharp\_am by default, preventing SHARP clients from communicating with sharp\_am. However, if you need to use UFM Appliance Gen 3.x with SHARP, you can resolve this by opening the required TCP port by running `ufw allow 6126/tcp`. Make sure that the port you specify in the 'smx\_sock\_port' config parameter matches the one you allow through the firewall.

---

# 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.

## 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_categories.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.

## 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.

## 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
```

---

# Operating NVIDIA SHARP in Dynamic Trees Allocation Mode

## SHARP Trees

A SHARP tree defines a set of switches and their connected links to be used by one or more SHARP jobs.

- A single tree can be used by multiple jobs, as long as they are using different areas of the tree.
- A single job can also utilize multiple trees, in case the job is operating on multiple rails, while each rail can use a different tree.

## Dynamic vs. Static Allocation Mode

In SHARP v3.3 and earlier, `sharp_am` used to operate in "Static trees" mode by default. In this mode, SHARP trees were created in the `sharp_am` initialization phase. When a new SHARP job started, it was assigned to one of the existing SHARP trees that was available to operate the job.

As of SHARP v3.4, `sharp_am`'s default operation mode is "Dynamic trees" mode. This mode is recommended as the preferred option to use.

When `sharp_am` operates in Dynamic trees mode, trees are not created in the initialization phase. Instead, they are created per job, immediately assigned to the job that requires them, and are deleted once the job ends.

The Dynamic trees mode of operation has some benefits over the Static trees mode, as it defines the SHARP configuration on the switches only when necessary, and enables better utilization of the fabric resource. There are various scenarios in which a Static mode of operation may respond with "No resources" to a SHARP job request, while in Dynamic mode, the SHARP job would be fulfilled.

# Dynamic Trees Allocation Algorithms

sharp\_am takes multiple factors into consideration when deciding on the trees to create for each job. Initially, the allocated trees must meet the job's requirements. However, sharp\_am also aims to allocate trees in a manner that preserves available links and switch resources for future jobs that may be needed.

The distinction between the combinations of trees that can be created in a regular FatTree versus a Quasi Fat Tree are significant. Consequently, sharp\_am offers two distinct algorithms to determine how trees should be created for each job. One algorithm is optimized for regular Fat-Tree fabrics, while the other is tailored for Quasi Fat Trees (QFTs) and Dragonfly fabrics.

Note the following:

- Only one algorithm can be used at a given time
- sharp\_am should be restarted when switching algorithms
- Under specific circumstances, when employing one algorithm and running multiple jobs simultaneously, sharp\_am might potentially declare "No resources" for a particular job request. However, if the other algorithm were utilized, the resources would be distributed differently, fulfilling all job requests.

Please note that there are no definitive right or wrong algorithms for any given topology, as each algorithm comes with its own advantages and limitations. Additionally, certain features are exclusive to specific algorithms.

It is recommended to consult with NVIDIA experts regarding the suitable algorithm for your system. You can contact us through either of the following methods:

E-mail: [Enterprisesupport@nvidia.com](mailto:Enterprisesupport@nvidia.com)

Enterprise Support page: <https://www.nvidia.com/en-us/support/enterprise>

## Configuring Dynamic Trees Allocation Mode

### Note

sharp\_am's default operation mode is set to Dynamic trees mode. Follow the instructions below in case you have the mode set to Static trees or a change of algorithm is required.

To operate in Dynamic trees mode, make sure `dynamic_tree_allocation` parameter is set to `TRUE`.

By default, the FatTree-oriented algorithm is used. To switch to the QFT-oriented or Dragonfly algorithm, use the `dynamic_tree_algorithm` parameter.

If the number of root switches in the fabric is larger than 126 when using the FatTree-oriented algorithm, it is desired to modify `max_trees_to_build` to be equal to the number of root switches.

Note that sharp\_am restart is required for the configuration to take effect.

## Limitations

- Dynamic trees allocation mode is currently available for Fat-Tree, Quasi-Fat-Tree (QFT), and Dragonfly topologies, and is not supported for Hypercube topologies. In case sharp\_am is configured to operate in Dynamic mode and the topology does not match, sharp\_am will automatically operate in Static mode.
- 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: <>
```

**Note:** You can avoid this warning by adding the following parameters to the ibdiagnet command line: `--sharp_opt ad_hoc`

- Dynamic trees creation 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.

---

# SHARP Reservation

Different entities, such as tenants, applications, jobs, and any desired group of nodes, can be bound together through SHARP reservation. In other words, SHARP reservation is a method for providing isolation and a set of attributes to each desired group of nodes.

In a public cloud system, this can be used to define a tenant. With the settings of a pkey (see [Operating NVIDIA SHARP with PKeys](#)), each tenant can run SHARP jobs for their applications. Different tenants' applications cannot impact one another, and the cloud admin can control which tenant can use SHARP. For simplicity, in this document, we address the use case of tenants and refer to the customer applications as running inside a tenant, although the reservation mechanism can serve other types of groups.

## Note

SHARP reservation feature is also called "SHARP allocation" in some APIs and configuration parameters.

To use the reservation capabilities, the following conditions should be met:

- sharp\_am should operate from within UFM, as UFM REST-API use is a must to operate in this mode.
- In UFM configuration file gv.cfg, the parameter enable\_sharp\_allocation must be set to True.

Once sharp\_am is in allocation mode, compute hosts cannot request a SHARP job unless it was specifically requested through the UFM REST-API.

The REST-API allows for the creation, updating, and deletion of tenants (reservations), with the option to define a related pkey and set a limit on the SHARP resources available to the tenant.

With this method, the fabric admin can control which compute hosts are allowed to leverage SHARP, and can even limit the number of trees allocated per tenant.

Full details of the REST-API can be found in the NVIDIA UFM Enterprise REST API guide.

## SHARP Resource Limit

Limiting the amount of SHARP resources per tenant is crucial in a multi-tenant system in order to prevent it from consuming resources and impacting other tenants.

To provide a fair use of the SHARP resources, the default configuration defines the following limits:

1. A tenant can run multiple SHARP jobs in parallel.
2. Two SHARP jobs cannot share the same HCA.
3. There is no explicit limit on the total number of SHARP jobs a tenant can run. However, since there is a limit per HCA and each SHARP job requires at least 2 HCAs, the effective limit on the total number of jobs is half the number of HCAs available to the tenant.

The limits described ensure a fair use of the resources and guarantee that, in a non-blocking topology, no tenant can allocate resources in a way that impacts another tenant's available resources.

You can set an explicit limit on the total number of jobs using the `app_resources_default_limit` configuration parameter (which applies to all tenants) or through the UFM REST-API (which allows different values for individual tenants).

Additionally, you can adjust the limit of jobs per HCA using the `reservation_max_jobs_per_hca` parameter, which affects all tenants.

---

# Operating NVIDIA SHARP with PKeys

SHARP can operate in a system that has either a single static special PKey, or a system that dynamically allocates PKeys.

## Defining a Special PKey

This method is used when SHARP is intended to operate exclusively on a single known PKey. To implement this, adjust the `ib_qpc_pkey` field to the desired PKey value in the SHARP configuration file.

Remember to ensure the membership bit is properly set, which entails setting both bit 0x8000 and the corresponding pkey value at all times.

## Supporting Dynamic PKeys

SHARP enables dynamic declaration of PKeys when operating from the UFM management host.

This feature is facilitated by the `enable_sharp_allocation` config parameter mentioned in the [SHARP Reservation](#) section. Configuring SHARP to operate in reservation mode via the UFM config file allows UFM to relay PKeys information to SHARP.

- To enable this functionality, make sure to set the following parameter in the UFM `gv.cfg` file:

```
sharp_allocation_enabled = true
```

- Restart UFM to apply the updated settings.

Note that the manipulation of PKeys does not automatically update the SHARP configuration, meaning that there are 2 sets of REST-API calls that need to be used:

1. The pure PKey REST API : This API affects the PKEY settings, however, it is not related to SHARP.

**(i) Note**

See the PKey GUIDs REST API section in the latest NVIDIA UFM Enterprise REST API guide [here](#).

2. The SHARP Reservation REST API: This API sets SHARP reservation and can refer to the pkeys that were created in the previous API.

**(i) Note**

See the NVIDIA SHARP REST API section in the latest NVIDIA UFM Enterprise REST API guide [here](#).

---

# 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.

---

# Testing NVIDIA SHARP Setup

## 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: <>
```

# 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
```

## NVIDIA SHARP Benchmark

The NVIDIA SHARP distribution provides source code for the benchmark, which tests native SHARP low-level performance for allreduce and barrier operations.

### Source code:

```
$module load hpcx
$HPCX_SHARP_DIR/share/sharp/examples/mpi/coll/
```

### Build and run instructions:

```
$module load hpcx  
$HPCX_SHARP_DIR/opt/Mellanox/sharp/share/sharp/examples/mpi/coll/R
```

---

# 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.

## NVIDIA SHARP Library Flags

### 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
<code>SHARP_LOG_VERBOSTIRY</code>	Log verbosity level 1 - Errors 2 - Warnings 3 - Info 4 - Debug 5 - Trace <b>Default:</b> 2
<code>SHARP_LOG_FILE</code>	Log file <b>Default:</b> stdout The log file name accepts the following modifiers in the file name to create a unique file <ul style="list-style-type: none"><li>• <b>%D</b> date as DDMMYYYY</li><li>• <b>%T</b> thread ID</li></ul>

Flag	Description
	<ul style="list-style-type: none"> <li>• <b>%H</b> host name</li> </ul>
SHARP_SMX SOCK_INTERFACE	<p>Network interface to be used by SMX: empty string (default) - Use interface used for AM connection <b>Default:</b> (null)</p>
SHARP_SMX SOCK_ADDR_FAMILY	<p>Determines which address family will be used in SMX's sockets. The value needs to be one of the following: { ipv4, ipv6, auto } <b>Default:</b> auto</p>
SHARP_SMX_UCX_INTERFACE	<p>Network interface to be used by SMX for UCX connections: empty string (default) - Use interface used for AM connection <b>Default:</b> (null)</p>

## 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	<p>Maximum payload per OST ( outstanding transactions) . Value 0 means "allocate default value". <b>Valid values:</b></p> <ul style="list-style-type: none"> <li>• 0 (default)</li> <li>• 128-1024</li> </ul>
SHARP_COLL_JOB_QUOTA_OSTS	<p>Maximum job (per tree) OST quota request. Value 0 means "allocate default quota". <b>Default:</b> 0</p>
SHARP_COLL_JOB_QUOTA_MAX_GROUPS	<p>Maximum number of groups (comms) quota request. Value 0 means "allocate default value".</p>

Flag	Description
	<b>Default:</b> 0
SHARP_COLL_OSTS_PER_GROUP	Number of OSTs per group. <b>Default:</b> 8
SHARP_COLL_JOB_QUOTA_MAX_QPS_PER_PORT	Maximum QPs/port quota request. Value 0 means "allocate default value".

## 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. <b>Default:</b> 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. <b>Default:</b> 16384
SHARP_COLL_SAT_LOCK_BATCH_SIZE	SAT lock batch size. Set this to "1" if multiple communicators want to use SAT resources. <b>Valid range:</b> 1-65535. <b>Default:</b> 65535 (Infinity)
SHARP_COLL_LOCK_ON_COMM_INIT	Get SAT Lock resource during

Flag	Description
	<p>communicator init if lock batch size is Infinity. Return failure if failed to lock</p> <p><b>Default:</b> 0 (Disabled), 1 (Enabled) with NCCL SHARP plugin</p>
<p>SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD</p>	<p>Lazy group resource allocation.</p> <p>0 - Disable lazy allocation, allocate group resource at communicator create time</p> <p>#n - Allocate sharp group resource after #n collective calls requested on the group</p> <p><b>Default:</b> 1</p>
<p>SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE</p>	<p>SAT (Streaming Aggregation Tree) exclusive lock mode for job.</p> <p>Possible values:</p> <ul style="list-style-type: none"> <li>• 0 - no exclusive lock</li> <li>• 1 - try exclusive lock</li> <li>• 2 (<b>default</b>)- force exclusive lock</li> </ul>
<p>SHARP_COLL_REDUCE_SCATTER_FRAG_SIZE</p>	<p>The fragment size of the reduce chunk in reduce-scatter collective operation.</p> <p><b>Default</b> - 512M</p>

# SHARP Miscellaneous Tuning

Flag	Description
SHARP_COLL_ENABLE_CUDA	Enables CUDA GPU support. Possible values: <ul style="list-style-type: none"> <li>• 0 - disable</li> <li>• 1 - enable</li> <li>• 2 <b>(default)</b> - try</li> </ul>
SHARP_COLL_PIPELINE_DEPTH	Size of fragmentation pipeline for larger collective payload. <b>Default:</b> 64
SHARP_COLL_ENABLE_MCAST_TARGET	Enables MCAST target on NVIDIA SHARP collective operations. Possible values: <ul style="list-style-type: none"> <li>• 0 <b>(default)</b> - disable</li> <li>• 1 - enable</li> </ul>
SHARP_COLL_MCAST_TARGET_GROUP_SIZE_THRESHOLD	Group size threshold to enable mcast target. <b>Default:</b> 2
SHARP_COLL_POLL_BATCH	Defines the number of CQ completions to poll on at once. <b>Valid range:</b> 1-16 <b>Default:</b> 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. <b>Default:</b> 180,000
SHARP_COLL_JOB_NUM_TREES	Number of SHARP trees to request. 0 means requesting the number of trees based on the number

Flag	Description
	of rails and the number of channels. <b>Default:</b> 0
<code>SHARP_COLL_GROUPS_PER_COMM</code>	Number of NVIDIA SHARP groups per user communicator. <b>Default:</b> 1
<code>SHARP_COLL_JOB_PRIORITY</code>	Job priority. <b>Valid values:</b> 0-10 <b>Default:</b> 0
<code>SHARP_COLL_ENABLE_PCI_RELAXED_ORDERING</code>	Enable PCI relaxed order memory access. Possible values: <ul style="list-style-type: none"> <li>• 0 - disable</li> <li>• 1 - enable</li> <li>• 2 (<b>default</b>) - auto</li> </ul>

**Note**

For the complete list of SHARP\_COLL tuning options, run the `sharp_coll_dump_config` utility:

```
$HPCX_SHARP_DIR/bin/sharp_coll_dump_config
```

---

# 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.

## HCOLL Library Flags

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

Flag	Description
<code>HCOLL_ENABLE_SHARP</code>	Sets whether SHARP should be used. Possible values: <ul style="list-style-type: none"><li>• 0 (<b>default</b>) – do not use NVIDIA SHARP</li><li>• 1 - probe NVIDIA SHARP availability and use it</li><li>• 2 - force to use NVIDIA SHARP</li><li>• 3 - force to use NVIDIA SHARP for all MPI communicators</li><li>• 4 - force to use NVIDIA SHARP for all MPI communicators and for all supported collectives (barrier, allreduce)</li></ul>
<code>SHARP_COLL_LOG_LEVEL</code>	NVIDIA SHARP coll logging level. Messages with a higher or equal level to the selected will be printed. Possible values: <ul style="list-style-type: none"><li>• 0 - fatal</li></ul>

Flag	Description
	<ul style="list-style-type: none"> <li>• 1 - error</li> <li>• 2 (<b>default</b>) - warn</li> <li>• 3 - info</li> <li>• 4 - debug</li> <li>• 5 - trace</li> </ul>
HCOLL_SHARP_NP	<p>Number of nodes (node leaders) threshold in the communicator to create NVIDIA SHARP group and use NVIDIA SHARP collectives.</p> <p><b>Default:</b> 4</p>
HCOLL_SHARP_UPROGRESS_NUM_POLLS	<p>Number of unsuccessful polling loops in libsharp coll for blocking collective wait before calling user progress (HCOLL, OMPI).</p> <p><b>Default:</b> 999</p>
HCOLL_ALLREDUCE_SHARP_MAX (or) HCOLL_BCOL_P2P_ALLREDUCE_SHARP_MAX	<p>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).</p> <p>The threshold is calculated based on the group resources.  <math>\text{Threshold} = \#OSTS * \text{Payload\_per\_ost}</math></p> <p><b>Default:</b> Dynamic</p>

## 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

---

# 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.

## 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 NVIDIA's Developer Guide for more details:

<https://docs.nvidia.com/deeplearning/sdk/nccl-developer-guide/docs/index.html>

## 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

## 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.

## 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/nvml-tests>

### Example:

```
$ mpirun -np 1024 -map-by ppr:8:node -x UCX_TLS=dc,shm,self -x
LD_LIBRARY_PATH=/sw/nvml/build/lib:~/sw/nvml-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
#

```

---

# 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.

## UCC Library Flags

The following HCOLL 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 <b>Note:</b> 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

## 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

---

# 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: <>"

## 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 -0  
/opt/ufm/files/conf/sharp/sharp_am.cfg --clean_and_exit TRUE
```

---

# 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
<code>metadata_host</code>	Hostname of the server running sharp_am.
<code>metadata_timestamp</code>	Unix timestamp (in seconds) indicating when data was generated; independent of request time.
<code>timestamp</code>	Unix timestamp (in milliseconds) showing when data was requested.
<code>active_jobs</code>	Total number of currently active SHARP jobs.
<code>active_sat_jobs</code>	Active SHARP jobs specifically requesting SAT rather than just LLT.
<code>agg_nodes_in_invalid_state</code>	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

SHARP telemetry is not active by default. To enable it, configuration changes are necessary. For assistance, it is recommended to contact [NVIDIA Support](#) to configure UFM to support SHARP telemetry.

---

# 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.

## 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 <dev:port> 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 vcheckdata directly from the HCA, the message size must be at least 16K.

## 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

```

## 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

```

## 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
```

## 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.

## 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
```

# Deployment Guide Revision History

Revision	Date	Description
3.10.2	December 24, 2024	N/A
3.9.0	November 07, 2024	<ul style="list-style-type: none"> <li>Added the following new sections:               <ul style="list-style-type: none"> <li><a href="#">Using NVIDIA SHARP with UCC</a></li> <li><a href="#">sharp_am Telemetry</a></li> <li><a href="#">HCA-to-TOR Benchmark Tool</a></li> </ul> </li> <li>Replaced most instances of MLNX_OFED with DOCA-Host through the document.</li> </ul>
3.8.0	August 14, 2024	<ul style="list-style-type: none"> <li>Modified SHARP Application Awareness title to SHARP Reservation; see <a href="#">SHARP Reservation</a></li> <li>Modified <a href="#">Operating NVIDIA SHARP in Dynamic Trees Allocation</a></li> <li>Modified <a href="#">SHARP Reservation</a></li> <li>Modified <a href="#">Operating NVIDIA SHARP with PKeys</a></li> </ul>
3.7.0	May 5, 2024	<p>In <a href="#">sharp_am Network Interfaces</a>, updated the following parameters:</p> <ul style="list-style-type: none"> <li><code>smx_</code> to <code>smx_sock_addr_family</code></li> <li><code>ucx_interface</code> to <code>smx_ucx_interface</code></li> </ul>
3.6.0	February 8, 2023	N/A
3.5.0	November 7, 2023	<ul style="list-style-type: none"> <li>Added <a href="#">sharp_am Log and Dump Files</a></li> <li>Updated <a href="#">NVIDIA SHARP Collective Library</a></li> </ul>
3.4.0	August 07, 2023	<ul style="list-style-type: none"> <li>Added <b>UFM Appliance Firewall Settings</b> under <a href="#">sharp_am Network Interfaces</a></li> <li>Updated <a href="#">Dynamic vs. Static Allocation Mode</a></li> <li>Updated <a href="#">Dynamic Trees Allocation Algorithms</a></li> <li>Added <a href="#">Operating NVIDIA SHARP with PKeys</a></li> <li>Under <a href="#">NVIDIA SHARP Collective Library</a>:</li> </ul>

Revision	Date	Description
		<ul style="list-style-type: none"> <li>◦ Updated <code>SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST</code> description</li> <li>◦ Updated the default value of <code>SHARP_COLL_OSTS_PER_G</code> flag</li> <li>◦ Updated <code>SHARP_COLL_SAT_LOCK_BATCH_SIZE</code> flag de</li> <li>◦ Added the default value for <code>SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_TH</code> flag</li> <li>◦ Updated the default value for <code>SHARP_COLL_ENABLE_MCAST_TARGET</code> flag</li> <li>◦ Updated <code>SHARP_COLL_POLL_BATCH</code> description</li> <li>◦ Added the default value for <code>SHARP_COLL_ERROR_CHECK_INTERVAL</code></li> <li>◦ Updated <code>SHARP_COLL_JOB_PRIORITY</code> flag description</li> <li>◦ Updated <code>SHARP_COLL_ENABLE_PCI_RELAXED_ORDERI</code> description</li> </ul>
3.3.0	May 1, 2023	<ul style="list-style-type: none"> <li>• Added <a href="#">Configuring Dynamic Trees Allocation Mode</a></li> <li>• Added <a href="#">SHARP Reservation</a></li> <li>• Updated <a href="#">Operating NVIDIA SHARP in Dynamic Trees Allocation</a></li> <li>• Updated <code>smx_sock_interface</code> entry under <b>Network Interfaces Configuration</b> table in <a href="#">sharp_am Network Interfaces</a></li> </ul>
3.1.1	November 30, 2022	Added <a href="#">NVIDIA SHARP Cleanup</a> .
3.1.0	October 31, 2022	<ul style="list-style-type: none"> <li>• Added <a href="#">Operating NVIDIA SHARP in Dynamic Trees Allocation M</a></li> <li>• Updated <a href="#">Modifying NVIDIA SHARP Aggregation Manager Conf</a></li> <li>• Updated Aggregation Trees Diagnostics under <a href="#">Testing NVIDIA Setup</a></li> </ul>
2.7.0	May 3, 2022	<ul style="list-style-type: none"> <li>• Updated Setup Requirements under <a href="#">Setting up NVIDIA SHARP Environment</a></li> <li>• Updated NVIDIA Hardware Capabilities and Limitations under <a href="#">Setting up NVIDIA SHARP Environment</a></li> <li>• Updated <a href="#">Running NVIDIA SHARP Aggregation Manager (AM) D</a></li> <li>• Updated <a href="#">Number of Simultaneous Streaming Aggregation Flow Disabling SHARP on Specific Network Devices in OpenSM</a></li> <li>• Added NVIDIA SHARP Configuration Flags under <a href="#">NVIDIA SHARP Collective Library</a></li> </ul>

Revision	Date	Description
		<ul style="list-style-type: none"> <li>Added NCCL_IBEXT_DISABLE under Control Flags in <a href="#">Using NVI SHARP with NVIDIA NCCL</a></li> </ul>
2.6.1	December 5, 2021	<p>Updated the following sections:</p> <ul style="list-style-type: none"> <li>Updated <a href="#">NVIDIA Hardware Capabilities and Limitations</a></li> <li>Updated the SHARP_COLL_ENABLE_SAT flag in <a href="#">NVIDIA SHARP Streaming Aggregation</a></li> <li>Updated SHARP variables and added NCCL SHARP Plugin variables <a href="#">Control Flags</a></li> </ul>

---

# Release Notes Revision History

## Release Notes Change History

### Changes and New Features History

Feature/Change	Description
Rev 3.9.0	
Enhanced Network Performance Benchmarking with SHARP	Added support for sharp_coll_test tool, which enables u benchmark the network performance between compute TOR switches effectively.
SHARP Client Errors Display in SHARP AM	MAD errors from SHARP clients are now displayed in SH facilitate monitoring of such events.
Bug Fixes	See <a href="#">Bug Fixes</a> .
Rev 3.8.0	
<b>DragonFly (DFP+) Topologies with Dynamic Trees Support</b>	Added support for DFP+ topologies while <code>sharp_am</code> or dynamic trees mode.
<b>SHARP Reservation Resource Limit</b>	Modified the logic of SHARP resource limit per reservati new parameter to control the number of jobs per HCA (s reservation_max_jobs_per_hca below). For further information, please see <a href="#">SHARP Reservation s</a>
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> .
Rev 3.7.0	
<b>Expanded SHARP Jobs Capacity</b>	Previously restricted to a maximum of 1023 simultaneou the capacity for SHARP jobs running concurrently has n enhanced. The new limit is determined by the size of the available switch resources
<b>Security QKey</b>	SHARP now supports the activation of a security QKey c nodes, ensuring a heightened level of security during op

Feature/Change	Description
<b>SHARP Telemetry Reports</b>	Added support for telemetry reports, delivering valuable specified intervals.
<b>Parameter Changes</b>	See <a href="#">Parameters Change History</a> below.
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> .
Rev 3.6.0	
<b>Parameter Changes</b>	smx_keepalive_refresh_interval smx_keepalive_min_time_before_connection_refresh smx_keepalive_min_percentage_of_connections_to_ref
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.5.1	
<b>New SHARP capability</b>	Added configuration parameters that control the desire regard to reservation scale-in and override of one reserv another.
<b>Parameter Changes</b>	load_reservation_files reservation_force_guid_assignment reservation_stop_jobs_upon_scale_in
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
<b>Rev 3.5.0</b>	
<b>Parameter Changes</b>	Added support for controlling the <code>sharp_am_log</code> mes level per the desired log category.
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.4.0	
<b>Parameter Changes</b>	dynamic_tree_allocation sharp_am A boolean parameter, indicates whether trees should be dynamically for each SHARP job or have trees allocated initialization. <b>Update:</b> Default value is now True
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.3.0	
<b>Syslog Capabilities</b>	Added support for a new syslog capability to libsharp. Syslog verbosity level can now be controlled using the SHARP_SYSLOG_VERBOSITY environment variable.

Feature/Change	Description
<b>Dynamic Trees Allocation Algorithms</b>	Added support for selecting one of two algorithms that trees should be created for each SHARP job. One algorithm for SuperPOD fabrics, while the other is optimized for Q (QFTs). For further information, please see <a href="#">Dynamic Trees Allocation</a> section.
<b>REST API Jobs Query</b>	Added support for retrieving the status of the current jobs along with the structure of the trees assigned to them. Note that this information is retrieved via REST-API and not of UFM.
<b>Unhealthy Ports</b>	Added support in OpenSM to inform SHARP of dangling links in order to avoid their use in SHARP jobs.
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.2.0	
<b>High Availability in sharp_am Network Interfaces</b>	sharp_am leverages multiple network interfaces of the remote host to provide high availability in case of a network interface failure. For further information, please see <a href="#">sharp_am Network Interfaces</a> section.
<b>Reliable Multicast</b>	Added support for SHARP to leverage reliable multicast on NVIDIA Quantum-2.
<b>SM Data</b>	Removed support for reading sm data by a client application. Functions sharp_request_sm_data, sharp_get_sm_data, and sharp_get_sm_data have been removed and can no longer be used. In addition, the configuration parameter ftree_ca_order has been removed from sharp_am.
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.1.1 LTS	
<b>SHARP Cleanup</b>	Added the ability to clean up all SHARP-related definitions and spare resources or to contribute to the recovery from an error state.
<b>General</b>	Updated MLNX_OFED and firmware versions in <a href="#">General</a> section.
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
Rev 3.1.0	
<b>Aggregation Manager (AM)</b>	Added support for dynamic creation of trees instead of static creation when SHARP is initialized.

Feature/Change	Description
Rev 3.0.1	
<b>Bug Fixes</b>	See <a href="#">Bug Fixes</a> section.
<b>Rev 3.0.0</b>	
<b>General</b>	Added support for executing multiple jobs that aggregate the same set of switches, while each job utilizes a different
	SHARP logic is now application-aware with UFM capabilities can be assigned an App-ID, which can be used as a reference for a 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.
<b>AM</b>	Modified the default resources provided to LLT & SAT jobs to support operation of a larger amount of SAT jobs in parallel to feature (please see the first three entries in the table below).
<b>libsharp</b>	SHARP jobs are now executed in exclusive lock mode by default (see SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE in the table below).
Rev 2.7.0	
<b>Switches</b>	Added support for NVIDIA Quantum-2 switches with ND
<b>Adapter Cards</b>	Added support for NVIDIA ConnectX-7 adapter card with 100GbE speed
<b>SHARPD</b>	sharpd daemon process has been removed. sharpd-related operations are now performed from the user application process
<b>AM</b>	Upon restart of AM, it no longer needs to wait for all containers to finish before being able to accept new jobs
	Added a mechanism that periodically checks for errors in job Trees and attempts to fix them
<b>General</b>	Added support for new data types BFLOAT16, INT8 and performing reduction operations
Rev 2.6.1	

Feature/Change	Description
<b>General</b>	Added support for running libsharp_coll from SHARP 2.6 from SHARP 2.4.0 – 2.6.1
<b>General</b>	Added information about updatable configuration parameters, configuration file and help menu
<b>Network</b>	Added support for keep-alive on connections to SHARPD
<b>Network</b>	Added support for asynchronous connections
<b>Network</b>	Disabled UCX listener as default in SHARP Aggregation Manager
<b>AM</b>	Added support for the non-default subnet prefix
<b>AM</b>	Added support for DF+ topologies with more than two-lanes
<b>SHARPD</b>	Added support for caching AM address
Rev 2.5.0	
<b>Resource Management</b>	Added support for exclusive lock requests for streaming jobs.
<b>Network</b>	Enabled connection keep-alive between SHARPD and Aggregation Manager.
Rev 2.4.3	
<b>General</b>	Added support for identifying Aggregation Nodes based on IP address
<b>General</b>	Improved minhop tables calculation.
<b>General</b>	Added a new API for querying events.
Rev 2.1.4	
<b>sharp_am/sharpd/libsharp_coll: Streaming Aggregation</b>	Added support for Streaming Aggregation over ConnectX-5 network card and Quantum switch.
<b>libsharp_coll: GPU Accelerator</b>	Added support for NVIDIA GPU buffers.
<b>sharp_am: OOB</b>	Added support for identifying the topology type from the SMDB file.
<b>sharp_am: Reboot</b>	Fixed an issue where recovery failed after reboot of all servers in a cluster.
Rev 2.0.0	
<b>sharp_am/sharpd/libsharp_coll</b>	Added support for the following NVIDIA Quantum switch features: <ul style="list-style-type: none"> <li>Performing data operations on new data types (unshort, and short floating point data types)</li> </ul>

Feature/Change	Description
	<ul style="list-style-type: none"> <li>1K OST payload</li> </ul>
<b>sharp_am/sharpd: Resource Management</b>	Added support for enabling and disabling reproducibility
<b>sharp_am/sharpd: Subnet Management</b>	Added support for controlling the SA key for SA operati
<b>libsharp_coll: GPUDirect</b>	Added support for CUDA GPUDirect and GPUDirect RDM
Rev 1.8.1	
<b>Aggregation Manager (sharp_am): Resiliency</b>	Added support for waiting for jobs to end prior to perform reinitialization on AM startup.
<b>Mellanox SHARP Daemon (sharpd): Out-of-Box Improvements</b>	Socket-based is now activated by default when installed RPM/MLNX_OFED.

## Parameters Change History

Parameter	Component	Description
<b>Rev 3.9.0</b>		
log_verbosity	sharp_am	Sets the sharp_am The default value is Info level.
ib_sa_key	sharp_am	Parameter is no longer Used to control the This information with the SM.
<b>Rev 3.8.0</b>		
reservation_max_jobs_per_hca	sharp_am	<b>New parameter:</b> A maximum number same HCA . A value of 0 means Valid range: 0-511 Applies only while <b>Default:</b> 1 job per
dynamic_tree_algorithm	sharp_am	Sets which algorithm tree mechanism.

Parameter	Component	Description
		Modified value 1 to support different network topologies. Current values: 0 - Regular FatTree 1 - Quasi Fat Tree
<b>Rev 3.7.0</b>		
rdma_sr_enable	sharp_am	<b>New parameter:</b> A boolean parameter that enables the rdmacm service, even when the rdma_sr_enable parameter is set to false. <b>Default:</b> True.
telemetry_interval	sharp_am	<b>New parameter:</b> A parameter that specifies the interval in seconds between telemetry updates. A value of 0 means that telemetry updates are disabled. Valid values: 0, 10-3600 <b>Default:</b> 60 seconds
telemetry_file_path	sharp_am	<b>New parameter:</b> A parameter that specifies the path of the telemetry log file. An empty path or a path that does not exist will result in an error. <b>Default in UFM:</b> /opt/ufm/log/telemetry.log <b>Default in non UFM:</b> /opt/ufm/log/telemetry.log
smx_sock_addr_family	sharp_am	Determines which address family to use for SMX's sockets. New option is added to support IPv6. Valid values: auto, ipv4, ipv6 The new "auto" option will use IPv4 or IPv6 depending on the configuration. If both IPv4 and IPv6 are configured, IPv6 can be used if it is configured. If only IPv4 is configured, IPv4 will be used. <b>Default:</b> auto.
SHARP_SMX_SOCKET_ADDR_FAMILY	libsharp	<b>Parameter Removed:</b> This environment variable is deprecated and will be removed in a future release. The parameter is replaced by the smx_sock_addr_family parameter, which is automatic, according to the configured address family.

Parameter	Component	Description
<code>SHARP_USE_USER_QKEY</code>	libsharp	<b>New parameter:</b> A libsharp should use user Qkey in case that a component security Qkey is not enabled and this environment variable is set to <code>true</code> . <b>Default:</b> False
<code>SHARP_SR_QUERY_SOURCE</code>	libsharp	<b>New parameter:</b> Used in order to fetch data from different sources. Possible values: 0 - Fetch only from the component supported options 1 - Fetch only from the component supported options and <code>sharp_am</code> is configured to <code>true</code> . 2 - Try both options. If not successful, try the component supported options. <b>Default:</b> 2 - Try both
Rev 3.5.0		
<code>log_categories_file</code>	<code>Sharp_am</code>	Added support for indicating the file path. The value "(NULL)" file does not exist. <b>Default:</b> In UFM, the path is <code>ufm/files/config</code>
Rev 3.3.0		
<code>dynamic_tree_algorithm</code>	sharp_am	<b>New parameter:</b> Used by the dynamic tree algorithm. This parameter is used to select the dynamic tree algorithm. Possible values: 0 - SuperPOD oriented 1 - Quasi Fat Tree <b>Default:</b> 0 - SuperPOD oriented
<code>app_resources_default_limit</code>	sharp_am	Sets the default number of resources used in parallel by the application. Modified the possible values. The value of -1 means unlimited.

Parameter	Component	Description
		resources by default. <b>Default:</b> -1 – No re
max_quota	sharp_am	<b>Deprecated parameter:</b> marked as deprecated; do not be used.
default_quota	sharp_am	<b>Deprecated parameter:</b> marked as deprecated; do not be used.
SHARP_SYSLOG_VERBOSITY	libsharp	<b>New parameter:</b> S level. Possible values: 0 – Disable syslog, 1 – Errors log level, 2 – Warnings log level, 3 – Info log level. <b>Default:</b> 1 – Errors
SHARP_GROUP_JOIN_MAD_TIMEOUT	libsharp	Sets the timeout for MAD in milliseconds. Modified the default value. <b>Default:</b> 3000 milliseconds
SHARP_GROUP_JOIN_MAD_RETRIES	libsharp	Sets the number of retries for MAD. Modified the default value. <b>Default:</b> 5 retries
SHARP_QP_CONFIRM_MAD_TIMEOUT	libsharp	Sets the timeout for confirmation MAD in milliseconds. Modified the default value. <b>Default:</b> 2000 milliseconds
Rev 3.2.0		
ignore_host_guids_file	sharp_am	<b>New parameter:</b> File path for ignored host GUIDs. <b>Default:</b> Null.
ignore_sm_guids	sharp_am	<b>New parameter:</b> Whether to ignore SM GUIDs in trees parsed from the network. <b>Default:</b> True.
ftree_ca_order_file	sharp_am	<b>Deprecated parameter:</b> marked as deprecated; do not be used.

Parameter	Component	Description
<code>enable_sat</code>	sharp_am	<b>Deprecated parameter:</b> Whether SHARP should use SAT. The parameter is ignored and should be set to TRUE. SAT is always supported.
<code>SHARP_COLL_SERIALIZE_MADS</code>	libsharp	<b>New parameter:</b> Should SHARP serialize MADS before connect and group. Recommended to be set to TRUE when using mpirun with multiple ranks. <b>Default:</b> False.
<code>SHARP_COLL_JOB_REQUEST_RMC</code>	libsharp	<b>New parameter:</b> If SHARP should use RMC for allocated SHARP trees. Multicast feature. <b>Default:</b> False.
<code>SHARP_COLL_FORCE_BCAST_AS_ALLREDUCE</code>	libsharp	<b>New parameter:</b> If SHARP should force bcast operation. <b>Default:</b> False.
Rev 3.1.1 LTS		
<code>clean_and_exit</code>	sharp_am	<b>New parameter:</b> If SHARP should clean and exit. If TRUE, sharp_am does not clean SHARP and exits. <b>Default:</b> False - Operator
Rev 3.1.0		
<code>dynamic_tree_allocation</code>	sharp_am	<b>New parameter:</b> If SHARP should allocate dynamic trees. All SHARP jobs should have dynamic trees. <b>Default:</b> False
<code>max_trees_to_build</code>	sharp_am	<b>Update:</b> In case dynamic trees are enabled (True), this parameter determines the maximum number of trees to build. The number of trees to build is determined by the number of possible trees. The number of possible trees is determined by the dynamic trees mode. The number of skeleton trees is recommended to be the number of ranks.

Parameter	Component	Description
		In case dynamic_t parameter can be <b>Default:</b>
SHARP_COLL_IB_TIMEOUT	libsharp	<b>New parameter:</b> T <b>Default:</b> 18
SHARP_COLL_IB_RETRY_COUNT	libsharp	<b>New parameter:</b> T <b>Default:</b> 7
SHARP_COLL_IB_RNR_TIMER	libsharp	<b>New parameter:</b> R <b>Default:</b> 12
SHARP_COLL_IB_RNR_RETRY	libsharp	<b>New parameter:</b> R <b>Default:</b> 7
SHARP_COLL_IB_SL	libsharp	<b>New parameter:</b> S <b>Default:</b> 0
SHARP_COLL_ENABLE_MCAST_TARGET	libsharp	<b>Update:</b> Modified False. <b>Default:</b> False
Rev 3.0.0		
per_prio_default_quota	sharp_am	<b>Update:</b> This parame percentage provid modified from 3 to
per_prio_default_sat_quota	sharp_am	<b>New parameter:</b> D Buffers and Group to be requested fo If no explicit quota parameter will set <b>Format:</b> prio_0_qu prio_9_quota] Note that if only o all priorities. <b>Default:</b> 3
sat_jobs_default_absolute_osts	sharp_am	<b>New parameter:</b> D allocated for SAT j Zero value means used, and the defa instead. Note that the num number of groups <b>Default:</b> 0

Parameter	Component	Description
<code>app_resources_default_limit</code>	sharp_am	<b>New parameter:</b> A only when reserva default max numb parallel by a single overridden per app. A value of 0 means an app can <b>Default:</b> 1
<code>force_app_id_match</code>	sharp_am	<b>New parameter:</b> A only when reserva to true, an applica request, and it mu provided upon res job will be denied. <b>Default:</b> False
<code>SHARP_COLL_JOB_REQ_EXCLUSIVE_LOCK_MODE</code>	libsharp	<b>Update:</b> Changed lock) to 2 (force ex
Rev 2.7.0		
<code>recovery_retry_interval</code>	sharp_am	<b>New parameter:</b> A recovery retries. A recover trees. <b>Default:</b> 300
<code>enable_seamless_restart</code>	sharp_am	<b>New parameter:</b> A to recover state fr operation of the c <b>Default:</b> True
<code>seamless_restart_trees_file</code>	sharp_am	<b>New parameter:</b> S Seamless restart. name, full path is c <b>Default:</b> sharp_am
<code>seamless_restart_max_retries</code>	sharp_am	<b>New parameter:</b> S retries of seamles more times in a ro run. <b>Default:</b> 3
<code>max_tree_radix</code>	sharp_am	<b>Update:</b> Change d
<code>Ib_sat_max_mtu</code>	sharp_am	<b>Update:</b> Change d that represents 4k

Parameter	Component	Description
per_prio_default_quota	sharp_am	<b>Update:</b> Changed more SAT jobs to switch.
Rev 2.6.1		
dump_dir	sharp_am	<b>Update:</b> Changed
smx_enabled_protocols	sharp_am	<b>Update:</b> Changed default)
ib_mad_timeout	sharp_am	<b>Update:</b> Change d
dump_dir	sharp_am	<b>Update:</b> Change d
sr_mad_timeout	sharpd	<b>New parameter:</b> C queries <b>Default:</b> 10000 mi
sr_mad_retries	sharpd	<b>New parameter:</b> C ServiceRecord que <b>Default:</b> 3 retires
<b>Rev 2.5.0</b>		
smx_keepalive_interval	sharp_am/sharpd	<b>New parameter:</b> K disable keep alive.l
smx_incoming_conn_keepalive_interval	sharp_am	<b>New parameter:</b> K connections <b>0 to c</b> <b>Default:</b> 300 seco
enable_exclusive_lock	sharp_am	<b>New parameter:</b> E feature. <b>Default:</b> True
enable_topology_api	sharp_am	<b>New parameter:</b> E <b>Default:</b> True
max_trees_to_build	sharp_am	<b>New parameter:</b> C build <b>Default:</b> 126
Rev 2.4.3		
ib_max_mads_on_wire	sharp_am	<b>Modified behavior</b> 4096

Parameter	Component	Description
ib_qpc_local_ack_timeout	sharp_am	<b>Modified behavior</b> 0x12
ib_sat_qpc_local_ack_timeout	sharp_am	<b>Modified behavior</b> 0x12
ib_qpc_timeout_retry_limit	sharp_am	<b>Modified behavior</b>
ib_sat_qpc_timeout_retry_limit	sharp_am	<b>Modified behavior</b>
Rev 2.0.0		
control_path_version	sharp_am	<b>New parameter</b> <b>Default</b>
max_compute_ports_per_agg_node	sharp_am	<b>Modified behavior</b> maximal radix valu <b>Default:</b> 0
default_reproducibility	sharp_am	<b>New parameter:</b> C for jobs. <b>Default:</b> TURE
ib_sa_key	sharp_am	<b>New parameter:</b> C <b>Default:</b> 0x1
coll_job_quota_max_payload_per_ost	sharp_job_quota	<b>Modified behavior</b>
SHARP_COLL_MAX_PAYLOAD_SIZE	Libsharp_coll	<b>Removed</b>
SHARP_COLL_NUM_SHARP_COLL_REQ	Libsharp_coll	<b>Removed</b>
SHARP_COLL_ENABLE_REPRODUCIBLE_MODE	Libsharp_coll	<b>New parameter:</b> C 0 – Use default. 1 – No reproducibi 2 – Reproducibility
SHARP_COLL_ENABLE_CUDA	Libsharp_coll	<b>New parameter:</b> E
SHARP_COLL_ENABLE_GPU_DIRECT_RDMA	Libsharp_coll	<b>New parameter:</b> E
Rev 1.8.1		
pending_mode_timeout	sharp_am	<b>New parameter:</b> D complete prior to
job_info_polling_interval	sharp_am	<b>New parameter:</b> D when waiting for j

# Bug Fixes History

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

Internal Ref.	Issue
4068969	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am
	<b>Discovered in Version:</b> 3.6.0
	<b>Fixed in Release:</b> 3.9.0
3844898	<b>Description:</b> 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.
	<b>Keywords:</b> SHARP, Report No resource
	<b>Discovered in Version:</b> 3.5.1
	<b>Fixed in Release:</b> 3.8.0
3438393	<b>Description:</b> 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.
	<b>Keywords:</b> Dynamic trees allocation; QFT; resource limitation
	<b>Discovered in Release:</b> 3.3.0
	<b>Fixed in Release:</b> 3.8.0
3971970	<b>Description:</b> 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.
	<b>Keywords:</b> Syslog
	<b>Discovered in Release:</b> 3.5.0
	<b>Fixed in Release:</b> 3.8.0

Internal Ref.	Issue
3478803	<b>Description:</b> Fixed the issue where obtaining topology information ( <code>sharp_cmd topology</code> ) failed when executed from the management host.
	<b>Keywords:</b> SHARP topology API
	<b>Discovered in Release:</b> 3.5.0
	<b>Fixed in Release:</b> 3.8.0
3844898	<b>Description:</b> 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.
	<b>Keywords:</b> SHARP, Report No resource
	<b>Discovered in Release:</b> 3.5.1
	<b>Fixed in Release:</b> 3.7.0
3696666	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am, libsharp, tcp/ip, smx
	<b>Discovered in Release:</b> 3.5.1
	<b>Fixed in Release:</b> 3.7.0
3686321	<b>Description:</b> When upgrading UFM from previous versions to UFM 6.15.x, <code>sharp_am</code> persistent directory as mentioned in the configuration file directs to a path that does not exist. This leads to a failure in saving reservation and job information, so in case of a restart of <code>sharp_am</code> , it won't be able to retrieve required information and return to its previous state.
	<b>Keywords:</b> <code>sharp_am</code> , UFM, upgrade
	<b>Discovered in Release:</b> 3.5.0
	<b>Fixed in Release:</b> 3.6.0
3724093	<b>Description:</b> 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.
	<b>Keywords:</b> <code>libsharp</code> , UCX

Internal Ref.	Issue
	<b>Discovered in Release:</b> 3.5.1
	<b>Fixed in Release:</b> 3.6.0
3665349	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am, libsharp
	<b>Discovered in Release:</b> 3.6.0
	<b>Fixed in Release:</b> 3.6.0
3646010	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am, Virtual Ports, OpenSM, Topology Policy
	<b>Discovered in Release:</b> 3.6.0
	<b>Fixed in Release:</b> 3.6.0
3609384	<b>Description:</b> Fixed issues concerning Sharp_AM connection creation with rank zero clients of active jobs during a restart when UCX is enabled.
	<b>Keywords:</b> sharp_am, libsharp, restart
	<b>Discovered in Release:</b> 3.4.0
	<b>Fixed in Release:</b> 3.5.0
3541153	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am, NCCL, UCX
	<b>Discovered in Release:</b> 3.4.0
	<b>Fixed in Release:</b> 3.5.0

Internal Ref.	Issue
3400293	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am; openSM
	<b>Discovered in Release:</b> 3.1.0
	<b>Fixed in Release:</b> 3.4.0
3479721	<b>Description:</b> Fixed the issue where sharp_am did not handle hypercube topologies well, causing it to incorrectly treat different switches as duplicates.
	<b>Keywords:</b> sharp_am; hypercube
	<b>Discovered in Release:</b> 3.3.0
	<b>Fixed in Release:</b> 3.4.0
3496440	<b>Description:</b> 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.
	<b>Keywords:</b> sharp_am
	<b>Discovered in Release:</b> 3.3.0
	<b>Fixed in Release:</b> 3.4.0
3336788	<b>Description:</b> Fixed the issue in Firmware where MAD error responses might have been received in libsharp.
	<b>Keywords:</b> sharp_am; libsharp
	<b>Discovered in Release:</b> 3.2.0
	<b>Fixed in Release:</b> 3.3.0 (Quantum-2 Firmware 31.2010.6064 )
3343503	<b>Description:</b> 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.
	<b>Keywords:</b> MLNX_OFED; sharp_am
	<b>Discovered in Release:</b> 3.2.0

Internal Ref.	Issue
	<b>Fixed in Release:</b> 3.3.0
3368381	<b>Description:</b> 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.
	<b>Keywords:</b> libsharp; MADs
	<b>Discovered in Release:</b> 3.0.0
	<b>Fixed in Release:</b> 3.3.0
3393902	<b>Description:</b> 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.
	<b>Keywords:</b> Virtual port; sharp_am; libsharp; SAT; ibv_poll_cq
	<b>Discovered in Release:</b> 3.2.0
	<b>Fixed in Release:</b> 3.3.0
3404474	<b>Description:</b> 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.
	<b>Keywords:</b> REST API; allocation
	<b>Discovered in Release:</b> 3.2.0
	<b>Fixed in Release:</b> 3.3.0
3406186	<b>Description:</b> Fixed an issue where SHARP AM failed handling reports from OpenSM if some switch ports were down or isolated.
	<b>Keywords:</b> Aggregation Manager; Aggregation Node; OpenSM
	<b>Discovered in Release:</b> 3.2.0
	<b>Fixed in Release:</b> 3.3.0
3236363	<b>Description:</b> 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.
	<b>Keywords:</b> Aggregation Manager; sharp_am; Link Failure
	<b>Discovered in Release:</b> 3.1.0
	<b>Fixed in Release:</b> 3.2.0

Internal Ref.	Issue
3230585	<p><b>Description:</b> 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.</p>
	<p><b>Keywords:</b> Dynamic tree; ibdiagnet</p>
	<p><b>Discovered in Release:</b> 3.1.0</p>
	<p><b>Fixed in Release:</b> 3.2.0</p>
3226743	<p><b>Description:</b> 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.</p>
	<p><b>Keywords:</b> Aggregation Manager; sharp_am; leaf; GUID</p>
	<p><b>Discovered in Release:</b> 3.0.1</p>
	<p><b>Fixed in Release:</b> 3.2.0</p>
3274564	<p><b>Description:</b> Fixed an issue where sharp_benchmark bash script failed to operate on all bash versions.</p>
	<p><b>Keywords:</b> sharp_benchmark</p>
	<p><b>Discovered in Release:</b> 3.1.1</p>
	<p><b>Fixed in Release:</b> 3.2.0</p>
3262936	<p><b>Description:</b> Fixed the issue where a crash took place during sharp_am reboot while physical links were hanging between switches in the fabric.</p>
	<p><b>Keywords:</b> sharp_am; physical links; crash</p>
	<p><b>Discovered in Release:</b> 3.1.0</p>
	<p><b>Fixed in Release:</b> 3.1.1 LTS</p>
3192770	<p><b>Description:</b> Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.</p>
	<p><b>Keywords:</b> SR-IOV</p>
	<p><b>Discovered in Release:</b> 3.0.0</p>

Internal Ref.	Issue
	<b>Fixed in Release:</b> 3.1.0
3163697	<b>Description:</b> 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.
	<b>Keywords:</b> File descriptor; libsharp; HCOLL; HPC-X
	<b>Discovered in Release:</b> 3.0.0
	<b>Fixed in Release:</b> 3.1.0
3192770	<b>Description:</b> Fixed the issue where SHARP jobs failed when using virtual interfaces configured with SR-IOV.
	<b>Keywords:</b> SR-IOV
	<b>Discovered in Release:</b> 3.0.0
	<b>Fixed in Release:</b> 3.0.1
3163697	<b>Description:</b> 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.
	<b>Keywords:</b> File descriptor; libsharp; HCOLL
	<b>Discovered in Release:</b> 3.0.0
	<b>Fixed in Release:</b> 3.0.1
2995739	<b>Description:</b> Sharp_am daemon is no longer removed when performing rpm upgrade and is overridden instead.
	<b>Keywords:</b> Aggregation Manager; rpm
	<b>Discovered in Release:</b> 2.6.1
	<b>Fixed in Release:</b> 2.7.0
2972970	<b>Description:</b> Fixed the issue where completion of SHARP installation using sharp_daemons_setup.sh script depended on python availability.
	<b>Keywords:</b> Aggregation Manager
	<b>Discovered in Release:</b> 2.6.1
	<b>Fixed in Release:</b> 2.7.0

Internal Ref.	Issue
2749073	<b>Description:</b> SHARP AM reports the rediscovery of aggregation nodes on every topology change.
	<b>Keywords:</b> Aggregation Manager
	<b>Workaround:</b> N/A
	<b>Discovered in Release:</b> 2.5.0
2736102	<b>Description:</b> SHARP AM and SHARPD overrides backlog files after restart when log rotation is enabled.
	<b>Keywords:</b> Aggregation Manager, SHARPD, log file
	<b>Workaround:</b> N/A
	<b>Discovered in Release:</b> 2.5.0
2700530	<b>Description:</b> 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.
	<b>Workaround:</b> N/A
	<b>Keywords:</b> SHARPD, Aggregation Manager
	<b>Discovered in Release:</b> 2.5.0
2726821	<b>Description:</b> Terminating SHARPD while the job process is still running will result in job resource leakage in SHARP Aggregation Manager.
	<b>Workaround:</b> Terminate SHARPD after terminating the job processes.
	<b>Keywords:</b> SHARPD, Aggregation Manager
2795902	<b>Description:</b> SHARPD might allocate handlers on GPU when running with UCX.
	<b>Keywords:</b> SHARPD, SMX, UCX
	<b>Workaround:</b> N/A
	<b>Discovered in Release:</b> 2.5.0
	<b>Workaround:</b> Disable UCX
2770210	<b>Description:</b> Syslog verbosity depends on log file verbosity.
	<b>Keywords:</b> SHARPD, Aggregation Manager
	<b>Discovered in Release:</b> 2.5.0
	<b>Workaround:</b> None

Internal Ref.	Issue
2825519	<b>Description:</b> Aggregation Manager continue to run after SM failover.
	<b>Keywords:</b> Aggregation Manager
	<b>Discovered in Release:</b> 2.5.0
	<b>Workaround:</b> Stop AM daemon manually
2754175	<b>Description:</b> SHARP Aggregation Manger might allocate bad links for jobs after receiving timeouts from Aggregation Nodes.
	<b>Workaround:</b> Restart corresponding switch or restart SHARP Aggregation Manager.
	<b>Keywords:</b> Aggregation Manager
	<b>Discovered in Release:</b> 2.5.0
2796317	<b>Description:</b> 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.
	<b>Workaround:</b> The PKEY used for creating the reservation should be "full" (the most significant bit should be on e.g. 0x805c instead of 0x5a).
	<b>Keywords:</b> Aggregation Manager, Reservations, PKEY, UFM
	<b>Discovered in Release:</b> 2.5.0

## Notice

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

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

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

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

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for

inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

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

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

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

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF 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.

## **Trademarks**

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

© Copyright 2025, NVIDIA. PDF Generated on 07/16/2025