NVIDIA GB200 and GB300 NVL 72 Partner Diagnostics

User's Guide

Document History

DU-11965-001_20

Version	Date	Description of Change
01	July 2024	Initial release.
02	August 2024	Added information about the switch tray coverage.
03	September 2024	• Added information about QS test coverage for compute and switch trays. • Added information about the field diagnostics.
04	September 2024	Added information about the L11 rack coverage.
05	October 2024	• Updated the workflow and prerequisites for L11 rack diagnostics. • Updated the test coverage for the L10 compute tray.
06	November 2024	Updated the L10 test coverage.
07	December 2024	• Updated GFM topology instructions. • Added L11 field diagnostics modes of operation. • Added nvlink_mask instructions for L10 switch diag and L11 rack diag.
08	December 2024	• Updated prerequisite software requirements. • Added cluster config to the global arguments for L11.
09	February 2025	• Updated with latest partner diagnostics test coverage. • Added information about how to run multiple L11 diagnostics instances from the same primary node. • Added Prometheus telemetry for switch node instructions. • Added IST instructions.
10	March 2025	• Updated with the latest field diagnostics tests and durations. • Added an example to set up a TCP IP bridge for L11 rack diagnostics.
11	March 2025	• Added documentation on C2CEgmDisabled global arg. • Added cable_cartridge test information.
12	March 2025	• Added note for overriding the L10 compute tray field diagnostics default behavior of expecting Gen 4 GPU. • Added note for modifying the BaseboardsPciIds in the field diagnostics spec json file.
13	June 2025	• Updated the IST section to align with v1.0. • Updated the ARM_FFA installation guidance. • Added OS dependencies. • Added GB300 information.
14	June 2025	• Updated test durations • Added run_nvlink_in_rack and nvlink_default_setting descriptions to global args
15	June 2025	Updated test durations
16	June 2025	• Added a numactl dependency. • Added a caution note to review the test spec before you run the partner diagnostics.
17	July 2025	• Updated ibstress test information. • Updated dependencies. • Added L11 execution recommendation note.
18	August 2025	• Added TegraCper test information • Updated supported arguments for powersync • Added environmentcheck test information

19	October 2025	• Fixed error_checking_level in ibstress • Added --failure_summary, --extra_partner_logging, and --partner_extra_logging_ms argument descriptions
20	November 2025	• Updated NVOnline ID for GB300 IST • Updated helpful arguments to specify '**=**' when required.

Contents

Chapter 1. Introduction

This guide provides information about how to use the NVIDIA® Partner Diagnostic software for NVIDIA GB200 and GB300 NVL 72 2:4 platforms for the compute tray, the switch tray, and the L11 rack.

NVIDIA Partner Diagnostic is a powerful software program that is used to qualify and factory test the NVIDIA partner design to isolate hardware failures. The Partner Diagnostics package includes support for Partner Manufacturing and Field Diagnostics.

1.1 About the Software Package

The software is provided as a tarball file, and the file must be extracted and executed on the system under test. The switch tray diagnostics must be executed in the NVOS environment on the switch tray, and the compute tray diagnostics must be executed in the host OS.

- Partner Diagnostics

  This is a suite of tests to isolate failures in the hardware.

  It comes as a tgz package with the following naming scheme:

  - For GB200 platforms:
    `629-PPPPP-KKKK-FLD-DDDDD.tgz` where:
    - `PPPPP` is the product code.
    - `KKKK` is the SKU code.
    - `DDDDD` is used for NVIDIA internal software tracking.

Chapter 2. Prerequisites

This chapter provides information about some simple tests that verify that the system is ready to run Manufacturing Diagnostics.

2.1 Aligning to the Latest Recipe

Before running the L11 rack partner diagnostics, users should complete the following steps:

1. Update the firmware and software on all compute trays and switch trays to the recommended recipe for the compute and switch tray diagnostics package.
2. Run the compute tray and switch tray manufacturing diagnostics.
3. Resolve issues in the compute and switch tray manufacturing diagnostics.
4. Update the firmware and software on all compute trays and switch trays to the recommended recipe for the L11 rack diagnostics.

> ⚠ **Warning:** This recipe might not match the recipe that was flashed in step 1.

5. Refer to *Cold Boot Sequence* in the *NVIDIA GB200 NVL System Bring-up Guide* and AC power cycle the rack.

2.2 Setting up the GFM Topology

1. Ensure that the GFM topology is configured correctly for the system by entering the appropriate topology option(s) in the `fm_config` file.
2. Confirm that the nvm-controller status is OK on the switch node that was set up.

Refer to the *NVIDIA GB200 NVL System Bring-up Guide* for more information.

2.3 Switch Level Checks

Validate each step on each switch tray.

1. To validate that nmx-c and nmx-t are running only on one of the switches, complete the following tasks:
   a. Verify that the status for each application is `ok`.
   ```
   admin@nvos:~$ nv show cluster apps running
   Name            Status  Reason
   --------------  ------  ------
   nmx-controller  ok
   nmx-telemetry   ok
   ```
   b. On switches that do not run cluster applications, run the following command.
   ```
   admin@nvos:~$ nv show cluster apps running
   No Data
   ```
2. Switch to the tray health and verify that no issues are observed.
   ```
   admin@nvos:~$ nv show system health
              operational  applied
   ```

```
----------  -----------  -------
status      OK
status-led  green



Health issues
================
No Data
```

3. Switch to the platform component, and on each switch, validate that all platform components are ok.

```
admin@nvos:~$ nv show platform environment
Name                              Type         State
--------------------------------  -----------  -----
ASIC1                             temperature  ok
ASIC2                             temperature  ok
Ambient-MNG-Temp                  temperature  ok
CPU-Pack-Temp                     temperature  ok
Drive-Temp                        temperature  ok
FAN1/1                            fan          ok
…
SWB-ASIC1-PCB-Temp                temperature  ok
SWB-ASIC2-PCB-Temp                temperature  ok
UID                               led          off
```

4. Link to the diagnostics, verify that the status of each link is ok, and that there are no issues.

```
admin@nvos:~$ nv show interface link-diagnostics
Interface  Code  Status
---------- ----- ----------------------
acp1       0     No issue was observed
acp2       0     No issue was observed
acp3       0     No issue was observed
acp4       0     No issue was observed
…
acp37      0     No issue was observed
acp38      0     No issue was observed
acp39      0     No issue was observed
acp40      0     No issue was observed
acp41      2     No issue was observed
…
acp72      2     No issue was observed
```

2.4 Fabric Level Checks

1. Check that the links in the fabric are ok, where the link width is 2x, the link speed is 212.5 Gpbs, and the link is up.

```
admin@nvos:~$ sudo iblinkinfo
Switch: 0xfc6a1c0300f4c5c0 MF0;nvos-f4c4c0:N5110_LD/U2:
...
          0   37[  ] ==( 2X         212.5 Gbps Initialize/  LinkUp)==>       0
2[  ] "GB100 Nvidia Technologies" ( Could be 2.5 Gbps)
          0   38[  ] ==( 2X         212.5 Gbps Initialize/  LinkUp)==>       0
```

```
2[  ] "GB100 Nvidia Technologies" ( Could be 2.5 Gbps)
          0   39[  ] ==( 2X         212.5 Gbps Initialize/  LinkUp)==>       0
2[  ] "GB100 Nvidia Technologies" ( Could be 2.5 Gbps)
          0   40[  ] ==( 2X         212.5 Gbps Initialize/  LinkUp)==>       0
...
CA: GB100 Nvidia Technologies:
     0x8f7f1dd473fee161      0    1[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   40[  ] "MF0;nvos-f4c4c0:N5110_LD/U1" ( Could be 2.5 Gbps)
     0x8f7f1dd473fee162      0    2[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   40[  ] "MF0;nvos-f4c4c0:N5110_LD/U2" ( Could be 2.5 Gbps)
CA: GB100 Nvidia Technologies:
     0x2925c6366cb693f7      0    1[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   39[  ] "MF0;nvos-f4c4c0:N5110_LD/U1" ( Could be 2.5 Gbps)
     0x2925c6366cb693f8      0    2[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   39[  ] "MF0;nvos-f4c4c0:N5110_LD/U2" ( Could be 2.5 Gbps)
CA: GB100 Nvidia Technologies:
     0x358fe2ef1a6a23fe      0    1[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   38[  ] "MF0;nvos-f4c4c0:N5110_LD/U1" ( Could be 2.5 Gbps)
     0x358fe2ef1a6a23ff      0    2[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   38[  ] "MF0;nvos-f4c4c0:N5110_LD/U2" ( Could be 2.5 Gbps)
CA: GB100 Nvidia Technologies:
     0x8c5f163cfb25be41      0    1[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   37[  ] "MF0;nvos-f4c4c0:N5110_LD/U1" ( Could be 2.5 Gbps)
     0x8c5f163cfb25be42      0    2[  ] ==( 2X         212.5 Gbps Initialize/
LinkUp)==>       0   37[  ] "MF0;nvos-f4c4c0:N5110_LD/U2" ( Could be 2.5 Gbps)
```

2.5 Software Prerequisites

This section provides a list of the software prerequisites for configurations and packages that need to be installed before you run partner diagnostics.

2.5.1 BIOS Configuration

Before running the partner diagnostics, MODS Secure partition (MODS SP) and Extended GPU Memory (EGM), and CPU Error Injection (EINJ) must be enabled on the system under test.

Note If you are using a non-reference UEFI, the process to enable MODS SP, EGM, and EINJ might differ.

On the NVIDIA reference UEFI, users must complete the following steps:

1. Reboot the system under test.
2. Enter the BIOS setup menu.
3. Navigate to Device Manager and click **NVIDIA Configuration** > **Grace Configuration**.
4. Complete the following tasks:
   a. Select the **MODS Secure Partition** checkbox.
   b. Select the **EGM** checkbox.
   c. Set the EGM Hypervisor reserved memory to 0.
   d. Select the **Error Injection** checkbox.

5. Reboot the system.

2.5.2 ARM Firmware Framework

Before you run the tool, ensure that the Arm Firmware Framework version 1.0 (ARM-FFA) kernel module is available for compilation or is installed on the system. The partner diagnostics provide an automated compilation and insert the ARM-FFA kernel module when the path to the module source is provided to the `-arm_ffa_src` argument. If this argument is not provided, the default path it looks for is `/lib/modules/<kernel>/updates/dkms`

NVIDIA also provides the ARM-FFA module as a DKMS package for Ubuntu 22.04 and RedHat 9.

- To install the DKMS package on Ubuntu 22.04, refer to the *DKMS package for ARM FFA module version 1.0 (.deb)* (NVOnline: **1126663**).
  Here are instructions to install the ARM FFA DKMS Debian package:

```
sudo mv
/lib/modules/<your_kernel_version>/kernel/drivers/firmware/arm_ffa/ffa-module.ko{,.
orig}
sudo rmmod ffa-module
sudo dpkg -i ffa-module-dkms_<version>_all.deb
```

- To install the DKMS package for RHEL9, refer to the *DKMS package for ARM FFA module version 1.0 (.rpm)* (NVOnline: **1126664**).
  Here are instructions to install the ARM FFA DKMS Red Hat® Enterprise Linux (RHEL) package:

```
sudo mv
/lib/modules/<your_kernel_version>/kernel/drivers/firmware/arm_ffa/ffa-module.ko{,.
orig}
sudo rmmod ffa-module
sudo dnf install ffa-module-dkms-<version>.noarch.rpm
```

The ARM-FFA module needs the MODS Secure Partition to be enabled in SBIOS. The value to look for in BMC is usually `ModsSpEnable`. Sometimes, the value in the SBIOS is cached and stale, so ensure that you power cycle the BMC accordingly.

To enable the ARM-FFA module, remove the MODS module by running the following command:

```
sudo rmmod mods
```

Users should power cycle after installing the ffa-module to resolve any potential failures.

2.5.3 Netstat

To run the diagnostics results analyzer (DRA) in the partner diagnostic, ensure that the Linux netstat tool is installed on the system for L10 and L11 diagnostics.

To install netstat, run one of the following commands:

- **Ubuntu:**

```
sudo apt install net-tools
```

- **RedHat:**

```
sudo yum install net-tools
```

If users cannot install netstat, they can explicitly configure the DRA ports using the `--dra_server_port` and `--global_dra_server_port` arguments when running the diagnostics.

2.5.4 OS Dependencies

There are several OS dependencies the partner diagnostics require to be used. The following sections provide a list of which tools and commands that are required.

2.5.4.1 NVIDIA DOCA Tools

The following software is required to run partner diagnostics:

- `mlxlink`
- `mst`
- `mlxreg`
- `mlxfwmanager`
- `mlxconfig`
- `mlxi2c_bf`
- `ibdev2netdev`
- `ibstat`
- `flint`
- `ib_umad`
- `mget_temp`
- `ibnetdiscover`
- `ibportstate`
- `resourcedump`
- `mstdump`
- `ibv_devinfo`
- `ib_read_bw`
- `ofed_info`
- `opensm`

2.5.4.2 Linux Commands

The following Linux commands are required to run partner diagnostics:

- `/bin/echo`
- `/bin/cd`
- `/bin/sh`
- `/opt/mellanox/iproute2/sbin/ip`
- `/usr/bin/as`
- `/usr/bin/awk`
- `/usr/bin/basename`
- `/usr/bin/bash`
- `/usr/bin/cat`
- `/usr/bin/cpupower`
- `/usr/bin/cut`
- `/usr/bin/date`
- `/usr/bin/dirname`
- `/usr/bin/dmesg`
- `/usr/bin/env`

- /usr/bin/expr
- /usr/bin/find
- /usr/bin/flock
- /usr/bin/fio
- /usr/bin/iostat
- /usr/bin/gcc
- /usr/bin/getconf
- /usr/bin/grep
- /usr/bin/head
- /usr/bin/ibdiagnet
- /usr/bin/ib_read_bw
- /usr/bin/ib_write_bw
- /usr/bin/id
- /usr/bin/ipmitool
- /usr/bin/jq
- /usr/bin/killall
- /usr/bin/ld
- /usr/bin/ln
- /usr/bin/ls
- /usr/bin/lscpu
- /usr/bin/lshw
- /usr/bin/lsof
- /usr/bin/lspci
- /usr/bin/lsusb
- /usr/bin/make
- /usr/bin/mdevices_info
- /usr/bin/mget_temp
- /usr/bin/mget_temp_int
- /usr/bin/minit
- /usr/bin/mkdir
- /usr/bin/mknod
- /usr/bin/mlxlink_int
- /usr/bin/mst_ib_add
- /usr/bin/mstop
- /usr/bin/objcopy
- /usr/bin/ofed_info
- /usr/bin/ps
- /usr/bin/pwd
- /usr/bin/python3
- /usr/bin/readlink
- /usr/bin/rm
- /usr/bin/sed
- /usr/bin/setpci
- /usr/bin/sh
- /usr/bin/sha512sum
- /usr/bin/sleep
- /usr/bin/sort
- /usr/bin/ssh
- /usr/bin/sshpass
- /usr/bin/strings
- /usr/bin/systemctl
- /usr/bin/tail
- /usr/bin/taskset
- /usr/bin/tee
- /usr/bin/touch
- /usr/bin/udevadm
- /usr/bin/uname

- /usr/bin/uniq
- /usr/bin/wc
- /usr/bin/which
- /usr/bin/numactl
- /usr/lib/gcc/x86_64-linux-gnu/11/cc1
- /usr/sbin/depmod
- /usr/sbin/dmidecode
- /usr/sbin/ibdev2netdev
- /usr/sbin/ibnetdiscover
- /usr/sbin/ibportstate
- /usr/sbin/ibstat
- /usr/sbin/ifconfig
- /usr/sbin/lsmod
- /usr/sbin/modinfo
- /usr/sbin/modprobe
- /usr/sbin/opensm
- /usr/sbin/smpquery
- /usr/sbin/smartctl

2.5.4.2 Linux Libraries

The following Linux libraries are required to run partner diagnostics:

- /lib/aarch64-linux-gnu/libcrypt.so.1
- /lib/aarch64-linux-gnu/libpthread.so.0
- /lib/aarch64-linux-gnu/libdl.so.2
- /lib/aarch64-linux-gnu/libutil.so.1
- /lib/aarch64-linux-gnu/librt.so.1
- /lib/aarch64-linux-gnu/libm.so.6
- /lib/aarch64-linux-gnu/libc.so.6
- /lib/aarch64-linux-gnu/libgcc_s.so.1
- /lib/aarch64-linux-gnu/libibumad.so.3
- /lib/aarch64-linux-gnu/libstdc++.so.6
- /lib/aarch64-linux-gnu/libz.so.1
- /lib/aarch64-linux-gnu/libmlx5.so.1.25.56.0
- /lib/aarch64-linux-gnu/libcudart.so

Chapter 3. Executing Partner Diagnostics on the System

The compute tray, switch tray, and rack-level diagnostics packages are posted separately. Each diagnostics package contains a `partnerdiag` executable file, a `onediagfield.rXX.YY.tar` file, and JSON configuration files. The partner diagnostics packages might also contain additional files that are required to execute the diagnostics. Refer to Manufacturing Diagnostics Guidelines and Field Diagnostics Guidelines for more information about these files.

Here is some additional information:

- The switch tray and compute tray diagnostics must be executed on the OS of the tray that is being tested.
- The rack-level diagnostics must be executed from an x86 server on the same network as the rack under test.

Refer to Manufacturing Diagnostics Guidelines for more information about running manufacturing diagnostics. Refer to Field Diagnostics Guidelines for more information about running the field diagnostics.

There will be various progress messages, and a PASS/FAIL banner will appear when the test has been completed. Refer to Interpreting the Diagnostic Results for more information about interpreting these messages. You can interrupt the diagnostic by clicking **Ctrl + C** or using other operating system utilities. After interrupting Partner Diagnostics, users must power cycle before running the tests again.

Chapter 4. Manufacturing Diagnostics Guidelines

4.1 Running the Manufacturing Diagnostics

The purpose of partner manufacturing diagnostics is to perform system integration tests in the partner's factory.

Test spec and SKU JSON files will be provided with the following naming schemes:

- Test Spec: `spec_<product-name>_<optional-description>_partner_mfg.json`
- SKU: `sku_<product-name>_partner_mfg.json`

The manufacturing diagnostics test spec and SKU JSON files are configured based on the NVIDIA reference system configuration, and partners might have to modify these input JSON files to fit their system configuration and platform needs. Refer to Spec JSON File for information about configuring the test spec JSON file. Refer to SKU JSON File for more information about modifying the SKU JSON file.

The switch tray and L11 rack diagnostics packages have a container image with a `onediagmfgcontainer.rXX.YY` naming scheme. This container image must be in the same directory as the `partnerdiag` executable when the diagnostics are launched. The L11 rack diagnostics use another input JSON file to configure the global diag manager (GDM) and a topology file to specify the NVLink topology.

Caution: Do not modify the L11 rack diagnostics GDM configuration JSON or topology files unless directed by an NVIDIA representative.

In this version of the L11 rack partner diagnostics, users should run the L11 diagnostics for the switch trays to completion and then run the L11 diagnostics for the compute trays. To run the L11 diagnostics on the compute trays, users must configure a spec JSON file where the `hosts` key in `global_args` contains the IP addresses and configuration information for the compute trays and switch trays in the rack under test, and configure the `test_target_node_type` to `compute_only`. The same procedure should be used when configuring a spec JSON file for the switch trays, but users should configure the `test_target_node_type` to `switch_only`.

When users run multiple instances of the L11 diagnostics from the same primary node, they must copy the diagnostics into other directories, and each instance must be run from its own location. To ensure that the ports being used are unique, and avoid collisions on the port, users must also modify the gdm and imex port parameters in Table 4-3 for each diagnostics instance.

The following sample command runs the manufacturing diagnostics for the compute tray or the switch tray. Run the command with root access:

```
sudo ./partnerdiag --mfg --run_spec=<testspec file> --no_bmc
```

The following sample commands run the manufacturing diagnostics for the L11 rack:

```
sudo ./partnerdiag --mfg --run_spec=<rack spec for switch> --primary_diag_ip=<IP>
sudo ./partnerdiag --mfg --run_spec=<rack spec for compute> --primary_diag_ip=<IP>
```

Caution: Test virtual_ids might change or new tests might be introduced between partner diagnostics releases.

Before you run a new version, to ensure that no tests are accidentally skipped, review the test json files.

Note: NVIDIA recommends users to run all L11 compute tray tests in a single run of the partner diagnostics. Previous limitations requiring users to execute tests individually have been resolved.

Table 4-1. Helpful Arguments for Manufacturing

Option	Description
--run_spec=<spec.json>	Runs the diag based on the specified Spec JSON file.
--arm_ffa_src=<path>	Provides the path to the `arm_ffa` source directory. The path is typically `<linux_source_tree>/drivers/firmware/arm_ffa`. If the host OS does not natively have the `arm_ffa` module, for the diagnostic package to function correctly, use this argument.
--log <absolute_path>	The path where the OneDiag run logs will be generated.
--run_on_error	Runs the specified tests and does not stop on failures.
--no_bmc	Runs without any BMC-related tasks. The BMC retrieves information about the system configuration and sensor data. This option disables correct product identification and prevents sensor monitoring and incorrect firmware version detection.
--fail_on_first_error	Runs all tests and fails on the first failure.
--help	Prints a complete list of arguments for manufacturing and field modes.
--skip_tests=<virtual ID>	Skips the test with the virtual ID provided as an argument to this option. Multiple tests can be skipped if they are provided as a comma-separated list of virtual IDs without spaces.
--unskip_tests=<virtual ID>	Unskips tests with the virtual ID provided as an argument to this option where the give virtual ID has skip_test set to true in the test json.

Option	Description
	Multiple tests can be unskipped if they are provided as a comma-separated list of virtual IDs without spaces.
--test=<virtual ID>	Runs only the test with the virtual ID provided as an argument to this option. Multiple tests can be run if they are provided as a comma-separated list of virtual IDs without spaces.
--primary_diag_ip=<Primary Diag IP>	Specifies the IP address of the primary node from which the diagnostic is being executed. This argument is only available in the partner diagnostics for L11 rack.
--global_dra_server_port=<port>	Port on which the DRA listens (primary node) or to which it connects (secondary node).
--gdm_fd=<socket>	• For a primary node, the Unix/Abstract socket fd to which the GDM listens. • For a secondary node, the Unix/Abstract socket to which the GDM connects.. Is used in the managed mode, and, for abstract sockets, ensure that you include @ .
--num_nodes_to_connect=<number>	Total number of secondary nodes that need to connect to the primary and is while running in the managed mode in the primary node.
--dra_client_fd=<socket>	Unix/Abstract socket fd to which the DRA client running on the secondary node connects.
--failure_summary	Enable NVLink failure summary log This argument invokes failure summary script that generates NVLink failure summary logs. This script requires python 3.9 or later and the pandas library. It is recommended to install pandas as sudo so it is available at a system level
--partner_extra_logging=<logs>	Runs the partner diagnostic with power, thermal, voltage or clock logged. Logging interval can be modified using --partner_extra_logging_ms Example usage: --partner_extra_logging=thermal,clock
--partner_extra_logging_ms	Sets the logging interval in milliseconds of the telemetry logged in --extra_partner_logging. The default value is 500ms and minimum value is 100ms.

Note: When executing the switch tray and L11 rack partner diagnostics, NMX-C **must** be active.

To activate NMX-C, run the following commands:

```
nv action stop cluster app nmx-controller
nv action start cluster app nmx-controller
```

4.2 Spec JSON File

The Spec JSON file specifies the test items that need to be run in the Partner Diagnostic. The tests are executed in order, and the spec file contains the arguments that can be configured for each test.

Spec files are typically divided into the following sections:

- Global args: A section that specifies arguments across all tests.
- Actions: A test list with the arguments that are specific to each test.

4.2.1 Test-Specific Arguments

Each test has a `virtual_id` that specifies the unique name of the test, and its `action` specifies the underlying test that is executed. Multiple tests can use the same action, but each test must have a unique `virtual_id`.

All tests support a `timeout_sec` argument that can be optionally specified on the outermost level of the `action json`. This argument specifies the period that the Partner Diagnostic will wait for the test to finish before it automatically kills the process.

In this guide, only the tests in which you will most likely need to change the test-specific args will be covered, and the tests are listed by `action`, instead of `virtual_id`. This is because tests with the same `action` support the same arguments.

Table 4-2. Actions and Completion Times

Actions	Test Duration	Test Description	Pass/Fail Criteria	Applicable Products
Inventory	One minute	System-level check of components against expected versions.	Fails if the firmware version does not match.	• GB200 compute tray • GB200 switch tray
inforom	40 seconds	Checks the GPU inforom		• GB200 compute tray • GB300 compute tray
tegra_cpu TegraCpu	10 minutes (configurable)	Performs CPU diagnostics testing.	Fails if the CPU operation is not stable.	• GB200 compute tray • GB300 compute tray
tegra_memory TegraMemory	45 seconds	Saturates the system memory	Fails if unable to perform	• GB200 compute

Actions	Test Duration	Test Description	Pass/Fail Criteria	Applicable Products
		bus with the concurrent data traffic that was generated by High Speed Scrubbing and the CPU. It requires a MODS secure partition.	read/write/allocate transactions on all memory channels.	tray • GB300 compute tray
tegra_memory CpuMemorySweep	22 minutes minutes	Perform reads, writes, correctness checks for CPU memory. This requires a MODS secure partition.	Fails if the test cannot allocate read/write/allocate transactions on all memory channels.	• GB200 compute tray • GB300 compute tray
tegra_clink	40 seconds	CPU-CPU NVIDIA® NVLink™ Test.	Fails if the link quality thresholds are not met.	• GB200 compute tray • GB300 compute tray
ssd	Two minutes (configurable)	SSD Stress test.	Fails if the SSD does not meet the performance of the input specification.	• GB200 compute tray • GB300 compute tray
pcieproperties	One second	Verifies the PCIe connection properties.	Fail if the detected PCIe properties do not match the spec JSON file.	• GB200 compute tray • GB300 compute tray

Actions	Test Duration	Test Description	Pass/Fail Criteria	Applicable Products
chckoccurrences	Five seconds	Checks a log for occurrences of a string. Partners can add error strings to this test configuration. The provided specification checks dmesg for errors that occurred during partner diagnostic runtime.	Fails if the specified string is found.	• GB200 compute tray • GB300 compute tray
gpustress	4.5 minutes	GPU stress tests.	Fails if there is a CRC miscompare during computation.	• GB200 compute tray • GB300 compute tray
gpumem	One minute	GPU memory and interface (FBIO) tests.	Fails if the GPU memory is unstable.	• GB200 compute tray • GB300 compute tray
pcie	12.5 minutes	GPU PCIe bandwidth, speed switching, and eye diagram tests.	Fails if the GPU PCIe connection is unstable or cannot achieve required functions.	• GB200 compute tray • GB300 compute tray
thermal	11.5 minutes (configurable)	Thermal test.	Fails if the temperature exceeds the limitation.	• GB200 compute tray • GB300 compute tray • GB200 L11 Rack
connectivity	12 minutes	Validates that the electrical quality of NVLinks and PCIE link speeds/width match the POR.	Fails if the NVLink connection detects errors or is unstable.	• GB200 compute tray • GB200 switch tray • GB200 L11 Rack

Actions	Test Duration	Test Description	Pass/Fail Criteria	Applicable Products
nvlink	12 minutes	NVLink bandwidth and eye diagram tests.	Fails if the link quality thresholds are not met.	• GB200 compute tray • GB200 L11 Rack
ibstress	Two minutes (configurable)	Performs infiniband stress read and write operations and verifies the performance.	Fails if the expected bandwidth is not met.	• GB200 compute tray • GB300 compute tray
dpudiag	Five minutes (configurable)	Performs infiniband traffic stress, eye diagram, and thermal stress of ConnectX-7 and Bluefield-3 Devices	Fails if the ConnectX-7 or Bluefield-3 does not meet performance standards.	• GB200 compute tray • GB300 compute tray
powersync	2.5 minutes per frequency (configurable)	Performs a synchronous CPU and GPU power stress.	Fails if the temperature exceeds the limitation.	• GB200 compute tray • GB300 compute tray • GB200 L11 Rack
c2c	Two minutes	CPU-to-GPU NVLink Stress.	Fails if there is poor electrical quality of c2c link.	GB300 compute tray
cable_cartridge	1.5 minutes	Checks that the cable cartridge FRU slot information is correctly programmed.	Flags if the cable cartridge FRU slot information does not match the expected for the given tray location.	GB200 L11 Rack
environmentcheck	1.5 minutes	Verifies that the required permissions and tools are installed for all actions selected to be run.	The required dependencies and/or permissions for the selected actions are not met	• GB200 compute tray
tegra_cpu TegraCper	1.5 minutes	Collects and decodes the CPER errors stored in the Grace R/W SPI flash on the system under test.	The TegraCper test is informational, and will fail only if the provided ruleset file is invalid or the tool is unable to	• GB200 compute tray

Actions	Test Duration	Test Description	Pass/Fail Criteria	Applicable Products
			read the CPERs from the R/W SPI flash.	

- The test time for the compute tray is approximately two-and-a-half hours.
- The test time for the switch tray is approximately two minutes.
- The test time for NVL L11 Rack is approximately two hours.

4.2.2 global_args

- Purpose: Specifies the attributes of the system/devices under test in the partner manufacturing diagnostics.
- The configurable fields are listed in Table 4-3.

4.2.2.1 Prometheus Switch Telemetry Endpoint Setup

Optionally, when the diag is only run on compute trays, but the switch telemetry is still queried and pass/failed on, users can enable the Prometheus Metric Endpoint on the switches. Using the Prometheus switch telemetry requires modifications to the `global_args` and the individual tests. Refer to Table 4-16 or connectivity for more information about enabling Prometheus telemetry during those tests.

The Prometheus Metric Endpoint Telemetry exposes an HTTP endpoint to integrate with monitoring systems that work in poll mode and support CSV formats. The endpoint provides only the last data sample, so users cannot obtain past data.

Here is an example of an endpoint call from the switch:

```
curl http:/0.0.0.0:9352/csv/metrics
```

Here is an example of an endpoint call from a remote node:

```
curl http:/#switch_ip#:9352/csv/metrics
```

By default, the Prometheus Endpoint is configured to **always** use port 9352.

To enable the endpoint on the Switches that are accessible from the Primary node:

1. After starting nmx-controller as mentioned in Table 4-1, run the following command on the same switch node.

   ```
   nv action start cluster app nmx-telemetry
   ```

2. After enabling the endpoint, enable the diag querying of the endpoint by adding `enable_prometheus` to the supported test arguments and `prometheus_url` to the global args shown in Table 4-3.

Table 4-3. global_args

Argument	Valid values	Type	Purpose	Applicable Products
BaseboardsPcIids	GPU PCI Addresses in Domain:Bus:device.function format	Array of strings	Specifies the GPUs being tested in the partner diagnostics.	Compute tray

gpu_pci_ids_loc_info_map	{
	<pci_address>: {
	"logical_id": <id_#>,
	"sxm_id" : <sxm_id_#>
	}
	}
		Json object	Maps the GPU PCI address to a logical and SXM ID.	Compute tray

gpu_pci_ids_slot_mapping	{
	<pci_address>:
	"GPU<logical_id>_<pci_address>
	}
		Json object	Maps the GPU PCI address to a slot name.	Compute tray

rf_endpoints	{
	"CBC_Eeprom_list:[
	"<CBC_FRU_redfish_API_0>",
	"<CBC_FRU_redfish_API_1>",
	"<CBC_FRU_redfish_API_2>",
	"<CBC_FRU_redfish_API_1>"
	]
	}
	}
		Json object	The CBC_Eeprom_list key is a list of redfish URIs for each CBC FRU EEPROM on the system under test, starting with /redfish/v1/.	L11 Rack

cluster_cfg	{
	"cluster_type":
	"GB200_NVL_2_4_72x1",
	"test_target_node_type":
	"compute_only",
	"cluster_node_logins":{
	"switch_node":{
	"user": <NVOS username>,
	"passwd": <NVOS password>,
	"key_filename": <SSH Key path>
	},
	"compute_node":{
	"user": <Host OS username>,
	"passwd": <Host OS password>
	"key_filename": <SSH Key path>
	}
	}
	}
		Json object	The cluster_type key specifies the target rack configuration. Valid values include GB200_NVL_2_4_36x1 and GB200_NVL_2_4_72x. Other configurations will be added in future releases.
			• The test_target_node_type key specifies whether diags should execute tests against all of the hosts in the test spec, only the compute nodes, or only the switch nodes.
			• The switch_node key specifies the username and password for NVOS.
			• The compute_node key specifies the username and password for the compute tray host OS.
				L11 Rack

Argument	Valid values	Type	Purpose	Applicable Products
			• The `key_filename` key specifies a path to a private key file that can be used for SSH authentication instead of using a password. Valid key algorithms include `ssh-ed25519`, `ecdsa-sha2-nistp256`, `ecdsa-sha2-nistp384`, `ecdsa-sha2-nistp521`, `rsa-sha2-512`, `rsa-sha2-256`, `ssh-rsa`, and `ssh-dss`. To access the NVOS restful APIs, include the NVOS username and password for the inventory test on the GB200 switch tray.	
	`global_dra_server_fd`	string	Unix/Abstract socket fd on which the global dra server is running on and on which the primary node will listen.	L11 Rack
	`gdm_port`	integer	The port number to which the gdm listens or connects. • The port **cannot** be used by other processes. • When running multiple instances of the diagnostics from the same primary node, the `gdm_port` must be unique for each diagnostics instance.	L11 Rack
	`imex_port_num`	integer	The port number to which the imex listens or connects. • The port **cannot** be used by other processes.	L11 Rack

Argument	Valid values	Type	Purpose	Applicable Products
			• When running multiple instances of the diagnostics from the same primary node, the `imex_port_num` must be unique for each diagnostics instance.	
hosts	{ "hosts": <ip address | socketfd>:{ "node_type": <node type>, "host_id": <host id>, "rack_slot":<slot_number>, "rack_id": "<customer_rack_identifier>" }	Json object	`<ip address>` specifies the IP addresses of the compute tray host OS and the switch tray NVOS. This address should match the address that is read from the TCP/IP socket connection with the compute or switch tray. For host_id, we recommend that you list the trays in order starting from the bottom of the rack for both switches and for compute trays and start at 0 to easily determine the physical location of the failure. See TRAY_INDEX in Figures 4-1 and 4-2 for numbering examples `<socketfd>` is a unique identifier for each compute nodes, and this value should match the value of the `--gdm_fd` arg that is passed to each compute tray's secondary diag. The `node_type` key specifies the type of tray to which the IP address belongs. • The valid values are `compute_node` and `switch_node`. • To identify the tray, the `host_type` key provides the diagnostic tool with an ID.	L11 Rack

Argument	Valid values	Type	Purpose	Applicable Products
			`rack_slot` is now a required field, and **all** cluster nodes must be listed for the hostname reporting of peer devices to function. rack_slot must be numbered starting at 1 from the bottom tray location in the rack and increment as you move up the rack regardless of node type. See SLOT_NUMBER in Figures 4-1 and 4-2 for numbering examples. rack_id is required to distinguish between two host ids which have the same rack_slot in multi-rack configuration. rack_id is a customer designated rack identifier string value. Use the `test_target_node_type` key in `cluster_cfg` to specify if this spec should address all nodes, only switch nodes, or only compute nodes.	
prometheus_url	Url of Prometheus Telemetry Endpoint http:/#switch_ip#:9352/csv/metrics	String	Defines the HTTP endpoint that can be queried for switch telemetry.	L11 Rack
prometheus_thresholds	JSON object containing error thresholds.	JSON object	Thresholds for Prometheus failure. Users should not modify this field. To skip checking errors on the switch, the field should be completely removed.	L11 Rack
C2CEgmDisabled	True to disable EGM or false to keep EGM enabled. This value is false by default.	Boolean	Used to disable EGM for the C2C bandwidth test.	Compute tray

Argument	Valid values	Type	Purpose	Applicable Products
			If this argument is not used, EGM will be enabled by default.	
run_nvlink_testing_in_rack	True/False	Boolean	Used to enable running L10 nvlink tests in an L11 rack configuration.	Compute Tray
global_args	nvlink_default_setting	List of Strings	Add nvlink_default_setting argument to the global_args list to enable backwards compatibility with firmware earlier than 1.1.00 for GB200.	GB200 Compute Tray GB200 L11 Rack

Example: `global_args` for Compute Tray (L10)

```
"global_args": {
    "product_config": {
        "BaseboardsPciIds" : [
            [ "0010:01:00.0", "0012:01:00.0", "0000:01:00.0", "0002:01:00.0" ]
        ],
        "gpu_pci_ids_loc_info_map": {
            "0010:01:00.0" : {
                "logical_id": 0,
                "sxm_id": 1
            },
            "0012:01:00.0" : {
                "logical_id": 1,
                "sxm_id": 2
            },
            "0000:01:00.0" : {
                "logical_id": 2,
                "sxm_id": 3
            },
            "0002:01:00.0" : {
                "logical_id": 3,
                "sxm_id": 4
            }
        },
        "gpu_pci_ids_slot_mapping" : {
            "0010:01:00.0" : "GPU0_0010:01:00.0",
            "0012:01:00.0" : "GPU1_0012:01:00.0",
            "0000:01:00.0" : "GPU2_0000:01:00.0",
            "0002:01:00.0" : "GPU3_0002:01:00.0"
        },
```

```
    },
    “run_nvlink_testing_in_rack”:true,
    “global_args”:[
       “nvlink_default_setting”
    ]
}
```

Example: `global_args` for Switch Tray (L10)

```
"global_args": {
       "cluster_cfg": {
            "cluster_node_logins": {
                "switch_node": {
                    "user": "admin",
                    "passwd": "admin"
                }
            }
        },
        "EnableDeviceVerify": false
}
```

Example: `global_args` for L11 Rack Compute Trays

```
   "global_args": {
        "DiagType": "partner_mfg",
        "DiagInfo": {
            "DiagName": "GB200-NVL L11 Partner Manufacturing Diag",
            "RunLevel": ""
        },
        "cluster_cfg": {
            "cluster_type": "GB200_NVL_2_4_72x1",
            "gdm_port": 8765,
            "imex_port_num": 5080,
            "imex_ctrl_srv_port_num": 1020,
            "l1_domain_id": 8193,
            "l2_domain_id": 16384,
            "node_address_config": "MNNVL_NODE_ADDRESS_CONFIG",
            "cluster_node_logins": {
                "compute_node": {
                    "user": "nvidia",
                    "passwd": "nvidia"
                },
                "switch_node": {
                    "user": "admin",
                    "passwd": "nvidia"
                },
                "inter_switch_node": {
                    "user": "root",
                    "passwd": "root"
                }
            },
            "hosts": {
                "10.114.248.6": {
                    "node_type": "compute_node",
```

```
                    "host_id": "COMPUTE_NODE_0",
                    "rack_id": "RACK_0",
                    "rack_slot": 1,
                },
                "10.114.248.7": {
                    "node_type": "compute_node",
                    "host_id": "COMPUTE_NODE_1"
                    "rack_id": "RACK_0",
                    "rack_slot": 2,
                },
                 .
                 .
                 .
                }
            }
        },
        "prometheus_url": "http://10.115.17.102:9352/csv/metrics",
        "prometheus_thresholds": {
            "FecEffectiveBer Bit Error Rate" : 1e-25,
            "BLER1" : 1e-25,
            "PhySymbBer Bit Error Rate" : 1e-25,
            "LinkDowned Errors": 0,
            "FecTotalRawBer Bit Error Rate": 3e-6,
            "PlrRcvBlkUcrrErrors Errors": 0
        },
        “global_args”:[
           “nvlink_default_setting”
        ]

    },
```

Example: `global_args` for L11 Rack Switch Trays

```
"global_args": {
        "cluster_cfg": {
            "gdm_port": 8765,
            "l1_domain_id": 8193,
            "l2_domain_id": 16384,
            "node_address_config": "MNNVL_NODE_ADDRESS_CONFIG",
            "cluster_node_logins": {
                "switch_node": {
                    "user": "admin",
                    "passwd": "nvidia"
                }
            },
            "hosts": {
                "13.13.13.13": {
                    "node_type": "switch_node",
                    "host_id": "SWITCH_NODE_0"
                },
                "14.14.14.14": {
                    "node_type": "switch_node",
```

```
                "host_id": "SWITCH_NODE_1"
            },
                .
                .
                .
            "15.15.15.15": {
                "user": "admin",
                "host_id": "SWITCH_NODE_8"
            },
        }
    },
    "product_config" : {
        "C2CLinkMask" : "gb_c1_g2",
        "rf_endpoints" : {
            "CBC_Eeprom_list" : [
                "/redfish/v1/Chassis/CBC_0",
                "/redfish/v1/Chassis/CBC_1",
                "/redfish/v1/Chassis/CBC_2",
                "/redfish/v1/Chassis/CBC_3"
            ]
        }
    }
    "EnableDeviceVerify": false,
    "enable_mods_gdm": false,
    "enable_gfm_in_gdm": false,
    "EnableMultinodeNvlinkTraffic": false,
    "fm_topo_file": "Bianca_36x1_topology",
    "mods_execution_retry_count" : 5
}
```

4.2.3 Actions

This section provides information about the supported test actions and test configuration options in the partner diagnostics package, and not all test actions are supported on all products. Refer to Table 4-2 to determine whether an action applies to the device or system under test.

4.2.4 inventory

- Purpose: Checks the system configuration against the provided JSON file.
- The configurable fields are listed in Table 4-4.

Table 4-4. inventory

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	integer	Kills the test if the test time exceeds the `timeout_sec` value.

Argument	Valid values	Type	Purpose
json_file	relative filepath	String	Gives the diagnostic package a JSON file to verify the inventory.

Example spec:

```
{
      "virtual_id": "Inventory",
      "test_level": "Level0",
       "args": {
           "json_file": "sku.json"
      },
      "timeout_sec": 1320,
      "action": "inventory"
}
```

4.2.5 tegra_cpu TegraCpu

- Purpose: Performs diagnostics tests on the existing Grace CPUs.
- The configurable fields are listed in Table 4-5.

Table 4-5. tegra_cpu TegraCpu

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec`.
-testTimeSec	Time in seconds	string	Sets the length of the `TegraCpu` test. This value must be smaller than the `timeout_sec` value.

Example spec:

```
{
     "virtual_id": "TegraCpu",
     "args": {
          "timeout_sec": 1200,
          "args": [
            "-prod", "GB200-NVL 2:4 board PC", "-test", "[1, 2, 3]",
              "-testTimeSec", "600"],
      },
       "action": "tegra_cpu"
}
```

4.2.7 tegra_memory TegraMemory

- Purpose: Saturates the system memory bus with the concurrent data traffic that was generated by High Speed Scrubbing and CPU.

  It requires a MODS secure partition.
- The configurable fields are listed in Table 4-7.

Table 4-7. tegra_memory TegraMemory

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
-testTimeSec	Time in seconds	string	Sets the length of the `tegramemory` test. This value must be smaller than the `timeout_sec` value.

Example spec:

```
        {
            "virtual_id": "TegraMemory",
            "args": {
                "timeout_sec": 1200,
                "args": [
                    "-prod", "GB200-NVL 2:4 board PC",
                    "-test", "[6,7]",
                    "-testTimeSec", "600"
                ]
            },
            "action": "tegra_memory"
        }
```

4.2.8 tegra_memory CpuMemorySweep

- Purpose: Perform reads, writes, and correctness checks for CPU memory.

  It requires a MODS secure partition.
- There are no configurable fields for this test.

Example spec:

```
       {
            "virtual_id": "CpuMemorySweep",
            "args": {
                "timeout_sec": 21600,
                "args": [
                    "-prod", "GB200-NVL 2:4 board PC",
                    "-test", "12",
                    "-concurrentMemTestLoopTime", "3600",
                    "-loop", "5"
                ]
            },
            "action": "tegra_memory"
        },
```

4.2.9 tegra_clink

- Purpose: Performs CPU-CPU NVLink diagnostics tests.
- There are no configurable fields for this test.

Example spec:

```
        {
            "virtual_id": "TegraClink",
            "args": {
                "is_mfg": false,
                "args": [
                    "-prod", "GB200-NVL 2:4 board PC",
                    "-test","5"
                ]
            },
            "action": "tegra_clink"
        },
```

4.2.10 ssd

- Purpose: Stresses SSDs using FIO and IOSTAT tools.
- The configurable fields are listed in Table 4-8.

Table 4-8. ssd

Argument	Valid values	Datatype	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
logicaldrive	SSD device path	String	Name of the logical drive.
physicaldrive	Array of SSD device paths	Array of strings	Names of the physical drives that correspond to the specified logical drive.
mountpoint	filepath	String	Mountpoint of the logical drive.
fioresult_threshold	Integer	String	FIO threshold.
perform_write_on_physical_drives	True/False	Boolean	Allows you to write a file to the drive. This argument might cause a corruption on the boot drive.
check_raid_configuration	True/False	Boolean	Checks whether the SSD is in the raid configuration.
bandwidthKB/s	Integer	String	Minimum bandwidth threshold for the FIO tests.
numjobs	Number of jobs as an integer	String	Used for `numjobs` argument in FIO test.
iodepth	iodepth as an integer	String	Used for `iodepth` argument in FIO test.
blocksizeKB	Blocksize as an integer	String	Used for the `bs` argument in FIO.
use_existing_mountpoints	True/False	Boolean	Specifies whether the diagnostic should use the existing SSD mountpoint or remount the device

Argument	Valid values	Datatype	Purpose
			to the filepath specified in `mountpoint`.
Report_physicaldrives_count_mismatch	True/False	Boolean	Verifies that the number of physical drives on the system match the number of physical drives in the input JSON file.
fioruntime	runtime in seconds	Integer	Specifies the runtime of the FIO test.
iostatruntime	Runtime in seconds	Integer	Specifies the runtime of the `iostat` test.
fiosize	Size in MiB	String	Used for the `size` argument in FIO.
operations_mask	1 byte of hexadecimal	String	Determines which FIO test to run: • Bit 0: randwrite • Bit 1: write • Bit 2: randread • Bit 3: read
chosen_physicaldrives_mask	Single hexadecimal value	String	A bitmask that determines the physical drives to target in the SSD testing.
chosen_logicaldrives_mask	Single hexadecimal value	String	A bitmask that determines the logical drives to target in the SSD testing.

Example Spec:

```
{
            "virtual_id": "Ssd",
            "args": {
                "ssd_drive_mappings" :
                [
                    {
            "logicaldrive"                          :   "/dev/nvme0n1",
            "physicaldrives"                        :   [ "/dev/nvme0n1" ],
            "mountpoint"                            :   "/",
            "fioresult_threshold"                   :   "90",
            "check_raid_configuration"              :   false,
            "read"                                  :   {
                                                         "bandwidthKB/s" : "3000000",
                                                        "numjobs"       : "8",
                                                        "iodepth"       : "4",
                                                        "blocksizeKB"   : "128"
                                                                    },
            "write"                                 :   {
                                                         "bandwidthKB/s" : "2400000",
                                                        "numjobs"       : "8",
                                                         "iodepth"       : "32",
                                                         "blocksizeKB"   : "128"
                                                          },
          "randread"                            :   {
                                                         "IOPS"          : "480000",
```

```
                                                "numjobs"       : "16",
                                                "iodepth"       : "16",
                                                "blocksizeKB"   : "4"
        "randwrite"                             :   {
                                                "IOPS"          : "130000",
                                                "numjobs"       : "16",
                                                "iodepth"       : "16",
                                                "blocksizeKB"   : "4"
                                                 }
                            }
            ],
            "use_existing_mount_points"             : true,
            "report_physicaldrives_count_mismatch"  : false,
            "fio_runtime"                           : 60,
            "iostat_runtime"                        : 5,
            "fio_size"                              : "10%",
            "fio_nice"                              : 0,
            "operations_mask"                       : "0x0C",
            "chosen_physicaldrives_mask"            : "0x3",
            "chosen_logicaldrives_mask"             : "0x3",
            "json_output"                           : true,
            "timeout_sec": 300
        },
        "action": "ssd"
    }
```

4.2.11 pcieproperties

- Purpose: Verifies the PCIe connection properties.
- The configurable fields are listed in Table 4-9.

Table 4-9. chckoccurences

Argument	Valid values	Type	Purpose
vendor_id	String of the vendor ID for the device under test.	String	Specifies the vendor ID that should be used to check the discovered vendor ID. The test will fail if the specified vendor ID does not match the discovered vendor ID.
device_id	String of the device ID for the device under test	String	Specifies the vendor ID that should be used to check the discovered vendor ID. The test will fail if the specified device ID does not match the discovered device ID.
link_width	String of intended negotiated link width for the device under test Valid values: “x1”, “x2”, “x4”, “x8”, or “x16”	String	Specifies a link width to check the current link width. The test will fail if the specified width does not match the enumerated width.

Argument	Valid values	Type	Purpose
link_speed	String of intended link speed for the device under test. Valid values: “2.5GT/s”, 5GT/s”, “8GT/s”, “16GT/s”, or “32GT/s”	String	Specifies a link speed to check the current link speed. The test will fail if the specified link speed does not match the enumerated link speed.
BDFs	Array containing strings of PCI addresses..	Array of Strings	Specifies the PCI addresses to test the PCIe properties.

Example spec:

```
      {
           "virtual_id": "CxPcieProperties",
           "args": {
               "vendor_id": "15b3",
               "device_id": "1021",
               "link_width": "x16",
               "link_speed": "32GT/s",
               "BDFs": [
                   "0000:03:00.0",
                   "0002:03:00.0",
                   "0010:03:00.0",
                   "0012:03:00.0"
               ]
           },
           "action": "pcieproperties"
       }
```

4.2.12 chckoccurences

- Purpose: Checks a log for occurrences of a string.

  It provides dmesg specification checks for errors that occurred during partner diagnostic runtime.
- The configurable fields are listed in Table 4-10.

Table 4-10. chckoccurences

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
command	Terminal command	String	Linux command to search the output for a string match.
logfilepath	String of linux filepath	String	Specifies the log file to search for a string.
fail_string	Any string	string	If an occurrence is found, the string prints as the failure message.

Example spec:

```
{
        "virtual_id": "DmesgLogErrorCheck",
        "test_level": "Level0",
        "args": {
                 "keyword_regex":
                "(?i).*error.*", "command":
                "dmesg", "timeout_sec": 300,
                "fail_string" : "Found errors in dmesg"
       },
        "action": "chkoccurrences",
}
```

4.2.13 gpustress

- Purpose: Performs GPU stress tests.
- There are no configurable fields for this test.

Example spec:

```
{
      "virtual_id": "Gpustress",
      "test_level": "Level0",
      "args": {
      },
      "action": "gpustress"
}
```

4.2.14 gpumem

- Purpose: Performs GPU stress tests.
- There are no configurable fields for this test.

Example spec:

```
{
     "virtual_id": "Gpumem",
     "test_level": "Level0",
     "args": {
     },
     "action": "gpumem"
```

```
}
```

4.2.15 pcie

- Performs GPU PCIe bandwidth, speed switching, and eye diagram tests.
- There are no configurable fields for this test.

Example spec:

```
{
   "virtual_id": "Pcie",
   "test_level": "Level0",
   "args": {
      "sequential": {
         "args": [
             "pex_bw_link_speed_mask_to_test=0xF"
         ]
      },
      "multiinstance": {
          "args": [
          ]
      }
    },
    "action": "pcie"
}
```

4.2.16 thermal

- Purpose: Thermal test for GB200 with a GPU.
- The configurable fields are listed in Table 4-11.

Table 4-11. thermal

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
tegra_telemetry	True/False	Boolean	Logs the CPU telemetry.
gpu_test_time_s	Time in seconds	Integer	Controls the test duration.

Example:

```
{
           "action": "thermal",
           "virtual_id": "Thermal",
           "args": {
               "args": [
               ],
               "override_spec": true,
               "override_spec_name": "timbakeTestSpec_titania_bianca",
               "cpustress": {
                   "tool": "mods",
                   "mods_args": []
```

```
                },
                "skip_spec_violations_check": true,
                "timeout_sec": 5600,
                "tegra_telemetry": false,
                "gpu_test_time_s": 3600
            }
}
```

4.2.17 connectivity

- Purpose: Validates the electrical quality of NVLinks and PCIE link speeds/width match the POR.
- To run the connectivity test on the compute tray or switch tray, loopback cables are required.
- The configurable fields are listed in Table 4-12.

Table 4-12. connectivity

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
nvlink_mask	hexadecimal bitmask	string	Bitmask to enable testing on a link. When running L10 switch tray diagnostics and L11 rack diagnostics, the `nvlink_mask` in the `compute_node` and `switch_node` sections need to be edited based on the system and/or rack configuration being tested. The `nvlink_mask` will differ between the compute and switch nodes, and the supported configurations are listed in Table 4-13.
enable_prometheus	True/False Default is False	Boolean	Only applicable to L11 rack diagnostics. Query Switch Telemetry data from the Primary Node from the endpoint. For each test that is run, fail if the switch error counters are above the defined thresholds

Table 4-13. Supported Configuration nvlink_mask for L11 Rack Diagnostics

Configuration	compute_node nvlink_mask	switch_node nvlink_mask
36x1	0x3FFFF	0x000000000FFFFFFFFF
36x2	0x3FFFF	0xFFFFFFFFFFFFFFFFFF
72x1	0x3FFFF	0xFFFFFFFFFFFFFFFFFF
8x1	0x3	0x00000000000000FF
18C:1S	0x3	0xFFFFFFFFFFFFFFFFFF

The 8x1 configuration in Table 4-13 assumes that users are targeting the compute tray at index 0 and index 1, and switch tray at index 0 as per Figure 4-1 and Figure 4-2. Each compute tray corresponds to one hexadecimal value in the `switch_node nvlink_mask`, where the right most hex number belongs to the compute tray at index 0 and increments as you move left across the bit mask. When testing 8x1, the `compute_node nvlink_mask` will change similarly depending on the index of the switch tray.

Table 4-14. Examples for 8x1 for Different Compute Tray Indices

Compute Trays	switch_node nvlink_mask
0, 1	0xFF
0, 2	0xF0F
1, 2	0xFF0
0, 17	0xF0000000000000000F

Table 4-15. Examples for 8x1 for Different Switch Tray Indices

Switch Trays	compute_node nvlink_mask
0	0x3
1	0xC
2	0x30
7	0x30000

Figure 4-1. 36X1{2} Configuration: 2U Compute Tray

CHASSIS_PHYS_ SLOT_NUMBER: <1-18>	Topology_ ID	<COMPUTE|SWITCH>_ TRAY_INDEX: <0-8>	Tray Name	Bianca-2U, NVL36 Cartridge Front view of the rack			
18	81h	8	Compute Tray#9	GPU#0	GPU#1	GPU#2	GPU#3
17	81h	7	Compute Tray#8	GPU#0	GPU#1	GPU#2	GPU#3
16	81h	6	Compute Tray#7	GPU#0	GPU#1	GPU#2	GPU#3
15	81h	5	Compute Tray#6	GPU#0	GPU#1	GPU#2	GPU#3
14	81h	4	Compute Tray#5	GPU#0	GPU#1	GPU#2	GPU#3
13	81h	8	Switch Tray#9	QM3#1		QM3#2	
12	81h	7	Switch Tray#8	QM3#1		QM3#2	
11	81h	6	Switch Tray#7	QM3#1		QM3#2	
10	81h	5	Switch Tray#6	QM3#1		QM3#2	
9	81h	4	Switch Tray#5	QM3#1		QM3#2	
8	81h	3	Switch Tray#4	QM3#1		QM3#2	
7	81h	2	Switch Tray#3	QM3#1		QM3#2	
6	81h	1	Switch Tray#2	QM3#1		QM3#2	
5	81h	0	Switch Tray#1	QM3#1		QM3#2	
4	81h	3	Compute Tray#4	GPU#0	GPU#1	GPU#2	GPU#3
3	81h	2	Compute Tray#3	GPU#0	GPU#1	GPU#2	GPU#3
2	81h	1	Compute Tray#2	GPU#0	GPU#1	GPU#2	GPU#3
1	81h	0	Compute Tray#1	GPU#0	GPU#1	GPU#2	GPU#3
				CC#1	CC#2	CC#3	CC#4

36x1{2} Configuration : 2U Compute Tray

NVL36 Cable Cartridge

(Cold Aisle View)

Figure 4-2. 72X1 Configuration: 1U Compute Tray

CHASSIS_PHYS_ SLOT_NUMBER: <1-27>	Topology_ ID	<COMPUTE|SWITCH>_ TRAY_INDEX: <0-17>	Tray Name	Bianca-1U, NVL72 Cartridge Front view of the rack			
27	80h	17	Compute Tray#18	GPU#0	GPU#1	GPU#2	GPU#3
26	80h	16	Compute Tray#17	GPU#0	GPU#1	GPU#2	GPU#3
25	80h	15	Compute Tray#16	GPU#0	GPU#1	GPU#2	GPU#3
24	80h	14	Compute Tray#15	GPU#0	GPU#1	GPU#2	GPU#3
23	80h	13	Compute Tray#14	GPU#0	GPU#1	GPU#2	GPU#3
22	80h	12	Compute Tray#13	GPU#0	GPU#1	GPU#2	GPU#3
21	80h	11	Compute Tray#12	GPU#0	GPU#1	GPU#2	GPU#3
20	80h	10	Compute Tray#11	GPU#0	GPU#1	GPU#2	GPU#3
19	80h	9	Compute Tray#10	GPU#0	GPU#1	GPU#2	GPU#3
18	80h	8	Compute Tray#9	GPU#0	GPU#1	GPU#2	GPU#3
17	80h	8	Switch Tray#9	QM3#1		QM3#2	
16	80h	7	Switch Tray#8	QM3#1		QM3#2	
15	80h	6	Switch Tray#7	QM3#1		QM3#2	
14	80h	5	Switch Tray#6	QM3#1		QM3#2	
13	80h	4	Switch Tray#5	QM3#1		QM3#2	
12	80h	3	Switch Tray#4	QM3#1		QM3#2	
11	80h	2	Switch Tray#3	QM3#1		QM3#2	
10	80h	1	Switch Tray#2	QM3#1		QM3#2	
9	80h	0	Switch Tray#1	QM3#1		QM3#2	
8	80h	7	Compute Tray#8	GPU#0	GPU#1	GPU#2	GPU#3
7	80h	6	Compute Tray#7	GPU#0	GPU#1	GPU#2	GPU#3
6	80h	5	Compute Tray#6	GPU#0	GPU#1	GPU#2	GPU#3
5	80h	4	Compute Tray#5	GPU#0	GPU#1	GPU#2	GPU#3
4	80h	3	Compute Tray#4	GPU#0	GPU#1	GPU#2	GPU#3
3	80h	2	Compute Tray#3	GPU#0	GPU#1	GPU#2	GPU#3
2	80h	1	Compute Tray#2	GPU#0	GPU#1	GPU#2	GPU#3
1	80h	0	Compute Tray#1	GPU#0	GPU#1	GPU#2	GPU#3
				CC#1	CC#2	CC#3	CC#4

72x1 Configuration : 1U Compute Tray

NVL72 Cable Cartridge

(Cold Aisle View)

An example for L10 Compute Tray diagnostics:

```
    {
        "virtual_id": "Connectivity",
        "args": {
            "pex": {
                "gpuPcieLinks": [
                {
                    "depth": 0,
                    "speed": 16000,
                    "upstreamWidth": 1,
                    "downstreamWidth": 1
                }]
            },
```

```
            "powercable": {
                "skip_test": true
            },
            "nvlink" : {
            },
            "i2c": {
                "skip_test": true
            },
            "infiniband": false,
            "ssd": false
        },
        "action": "connectivity"
    }
```

An example for L10 Switch Tray diagnostics:

```
    {
        "virtual_id": "connectivity",
        "args": {
            "nvlink": {
                "args": [
                    "infiniband_ca_name=sx_ib_0",
                    "nvlink_mask=0x000000000FFFFFFFFF"
                ]
            },
            "powercable" : {
                "skip_test" : true
            },
            "pex" :{
                "skip_test" : true
            },
            "infiniband" : false,
            "timeout_sec": 982,
            "ssd": false,
            "i2c" : {
                "skip_test" : true
            }
        },
        "action": "connectivity"
    }
```

An example for L11 Rack diagnostics for 36x1:

```
    {
        "virtual_id": "Connectivity",
        "args": {
            "timeout_sec": 5400,
            "nvlink": {
                "args": [
                ],
                "compute_node": {
                    "args": [
                      “nvlink_mask=0x3FFFF”
                    ],
                    "run_mods_type": "sequential"
```

```
                },
                "switch_node": {
                    "args": [
                            "nvlink_mask=0x000000000FFFFFFFF"
                    ]
                }
            },
            "powercable": {
                "skip_test": true
            },
            "pex": {
                "skip_test": true
            },
            "i2c": {
                "skip_test": true
            },
            "ssd": false,
            "infiniband": false,
            "fnm_links": true,
            "enable_prometheus": false
        },
        "action": "connectivity"
    }
```

4.2.18 nvlink

- Purpose: GPU-GPU NVIDIA NVLink Test.
- To run the nvlink test on the compute tray, and loopback cables are required.
- The configurable fields are listed in Table 4-16.

Table 4-16. nvlink

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
enable_prometheus	True/False Default is False	Boolean	Only applicable to L11 rack diagnostics. Query Switch Telemetry data from the Primary Node from the endpoint. For each test that is run, fail if the switch error counters are above the defined thresholds

Here is an example:

```
    {
        "virtual_id": "Nvlink",
        "args": {
            "singleinstance": {
                "args": [
                ]
            },
            "concurrent_uva": {
```

```
                    "args": [
                    ]
                },
                "concurrent": {
                    "args": [
                    ]
                },
                "timeout_sec": 3600,
                "enable_prometheus": false
            },
            "action": "nvlink"
        }
```

4.2.19 ibstress

- Purpose: Performs infiniband stress read and write operations using the perftest suite and verifies link quality and performance by monitoring bandwidth, bit error rate, and temperature.

  This test can be used on ConnectX and Bluefield-3 devices.
- Running ibstress test on the Bluefield-3 devices requires the devices to be in DPU mode.

  By default, Bluefield-3 devices are in DPU mode.

  Refer to Modes of Operation for more information about configuring the Bluefield-3 devices to use the DPU mode.
- When running the ibstress test, ensure that the device ports are active.
  a. To check the state of the ports, connect to the Bluefield CPU OS, run the following commands.
  ```
  sudo opensm -o
  ibstat
  ```
  b. Verify that the port state is *Active* and the Link Status is *UP*.
- If you are using loopback cables (`test_connect_mode` of `cables` or `gpudirect` with cabled configuration), attach the cables between the physical ports on each ConnectX or Bluefield-3 card.
- The configurable fields are listed in Table 4-17.

Table 4-17. ibstress

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
test_mode	concurrent/sequential	string	Determines whether to concurrently or sequentially run the ibstress test on the devices.
test_connect_mode	phyloopback/cables	string	Determines whether to run the ibstress test using internal loopback or external loopback using cables.

Argument	Valid values	Type	Purpose
			When running in gpudirect mode, the `topology` field determines whether the device should be run in phyloopback or with cables.
skip_subnet_manager	true/false	Boolean	Determines whether to skip the subnet manager setup for RoCE/Ethernet mode.
use_packaged_perftest	true/false	Boolean	Determines if the partner diagnostics use the perftest packaged with the diagnostic or use the perftest present on the system.
topology	RMDA connection map	JSON object	Maps the RDMA devices to each other and how loopback cables are connected on the system. If no loopback cable is used, to perform an internal loopback, map an RDMA to itself. When running uni-directionally, users should specify topologies for both client/server cases, for example `“mlx5_0”:”mlx5_1”` and `“mlx5_1”:“mlx5_0”`. When running bi-directionally, only one client/server case is required.
expected_pcie_id_to_ib_dev_map	RDMA to PCIE BDF map	JSON object	Maps the RDMA devices to their BDF. Used for topology validation.
rdma_over_ethernet_config	Port name to IP and RDMA map	JSON object	Maps the port to its IP address and RDMA. Used only in RoCE/ETH mode. The IP addresses assigned in this field can be any IPv4 address, but the address must be different from any other IP address that is associated with the system under test. The ethernet interface name must match the port name that corresponds to the ib_dev seen in the output of `sudo mst status -v`. If the argument is not used, the IPs will be chosen by the partner diagnostics automatically.
device_numa_map	RDMA to numa map	JSON object	Maps the RDMA device to a numa node for affinity.
sub_tests	ib_read_bw, ib_write_bw	JSON object	Determines the ibstress subtest and test arguments.
expected_BW_GBps	Bandwidth in GB/s	integer	Bandwidth threshold against which the ibstress tests will be verified.

Argument	Valid values	Type	Purpose
duration_secs	Length in seconds	integer	The time to run the ibstress subtest.
ibdev_gpu_map	RDMA to GPU index map	JSON object	Maps the RDMA devices to the GPU index to which the ConnectX-8 is connected. This field is required for the gpudirect test mode.
bidirectional	True/False	Boolean	Determines whether ibstress should be run bi-directionally or uni-directionally. The default value is false.
errorl_checking_level	“all” or subset of “ber”, “bler”, “ bandwidth”, “temp”	List of strings	Determines the telemetry to be collected and validated for ConnectX-8 devices.
ping_timeout_sec	time in seconds	integer	Determines the amount of time before timeout for ping checks in seconds.
subnet_manager_timeout_sec	time in seconds	integer	Determines the amount of time before timeout for the subnet manager setup in seconds
link_layer_mode	IB, ETH	string	Specifies the mode of operation. If this field is not used, the partner diagnostics will default to ETH mode.

Example spec for ibstress using loopback cables:

```
      {
           "virtual_id": "IbStressCables",
           "test_level": "Level1",
           "args": {
               "test_mode": "sequential",
               "test_connect_mode": "cables",
               "topology": {
                   "mlx5_1": "mlx5_0",
               },
               "expected_pcie_id_ib_dev_map": {
                   "0000:03:00.0": "mlx5_0",
                   "0002:03:00.0": "mlx5_1"
               },
               "rdma_over_ethernet_config": {
                   "ibp3s0": {
                       "ip": "10.10.10.10",
                       "ib_dev": "mlx5_0"
                   },
                   "ibP2p3s0": {
                       "ip": "11.11.11.11",
                       "ib_dev": "mlx5_1"
                   }
               },
```

```
            "sub_tests": {
                "ib_read_bw": {
                    "expected_BW_GBps": 40,
                    "duration_secs": 5,
                    "base_server_port": 18515,
                    "server_port_offset": 100,
                    "server_setup_timeout_secs": 5
                },
                "ib_write_bw": {
                    "expected_BW_GBps": 40,
                    "duration_secs": 5,
                    "base_server_port": 18515,
                    "server_port_offset": 100,
                    "server_setup_timeout_secs": 5
                }
            }
        },
        "action": "ibstress"
    }
```

Example spec for ibstress gpudirect using external loopback cables:

```
    {
        "action": "ibstress",
        "virtual_id": "Cx8GpuDirectCrossGpu",
        "args": {
            "test_mode": "concurrent",
            "test_connect_mode": "gpudirect",
            "skip_subnet_manager": false,
            "use_packaged_perftest": true,
            "bidirectional": true,
            "topology": {
                "mlx5_0": "mlx5_2",
                "mlx5_1": "mlx5_3",
                "mlx5_2": "mlx5_0",
                "mlx5_3": "mlx5_1",
                "mlx5_4": "mlx5_6",
                "mlx5_5": "mlx5_7",
                "mlx5_6": "mlx5_4",
                "mlx5_7": "mlx5_5"
            },
            "ibdev_gpu_map": {
                "mlx5_0": "1",
                "mlx5_1": "1",
                "mlx5_2": "0",
                "mlx5_3": "0",
                "mlx5_4": "3",
                "mlx5_5": "3",
                "mlx5_6": "2",
                "mlx5_7": "2"
```

```
            },
            "rdma_over_ethernet_config": {
                "enp3s0f0np0": {
                    "ip": "11.136.187.244",
                    "ib_dev": "mlx5_0"
                },
                "enp3s0f1np1": {
                    "ip": "11.136.187.245",
                    "ib_dev": "mlx5_1"
                },
                "enP2p3s0f0np0": {
                    "ip": "11.136.187.246",
                    "ib_dev": "mlx5_2"
                },
                "enP2p3s0f1np1": {
                    "ip": "11.136.187.247",
                    "ib_dev": "mlx5_3"
                },
                "enP16p3s0f0np0": {
                    "ip": "11.136.187.248",
                    "ib_dev": "mlx5_4"
                },
                "enP16p3s0f1np1": {
                    "ip": "11.136.187.249",
                    "ib_dev": "mlx5_5"
                },
                "enP18p3s0f0np0": {
                    "ip": "11.136.187.250",
                    "ib_dev": "mlx5_6"
                },
                "enP18p3s0f1np1": {
                    "ip": "11.136.187.251",
                    "ib_dev": "mlx5_7"
                }
            },
            "expected_pcie_id_ib_dev_map": {},
            "sub_tests": {
                "ib_read_bw": {
                    "num_qps": 16,
                    "expected_BW_Gbps": 700,
                    "duration_secs": 10,
                    "base_server_port": 18515,
                    "server_port_offset": 100,
                    "server_setup_timeout_secs": 5
                },
                "ib_write_bw": {
                    "num_qps": 16,
                    "expected_BW_Gbps": 700,
                    "duration_secs": 10,
```

```
                    "base_server_port": 18515,
                    "server_port_offset": 100,
                    "server_setup_timeout_secs": 5
                }
            }
        }
    }
```

4.2.19.1 System Configuration for gpudirect

This section provides the configuration steps that are required to run the gpudirect mode for designs with ConnectX-8.

1. Before you run the test, ensure that the NVIDIA GPU driver and cuda-toolkit 12.8 or 12.9 is available on the system under test.
2. Add the following registries to /etc/modprobe.d/nvidia.conf
   ```
   options nvidia NVreg_GrdmaPciTopoCheckOverride=1
   NVreg_RegistryDwords="RmGpuFabricProbe=0x80000000;"
   ```
3. Use mlxconfig to configure the ConnectX-8 for either IB or Ethernet. You can find the device names by running the following command:
   ```
   sudo mst status -v
   ```

   To configure the device for IB:
   ```
   sudo mlxconfig -d <device> set LINK_TYPE_P1=1
   ```

   To configure the device for Ethernet:
   ```
   sudo mlxconfig -d <device> set LINK_TYPE_P1=2
   ```

4. Configure ConnectX-8 ports splits..

   For **Infiniband mode**, the default ConnectX-8 setting is 1x800G port in Infiniband mode, run the following command to reset the device to Infiniband mode:
   ```
   sudo mlxconfig -d <device> reset
   ```

   For **ethernet mode,** put each ConnectX-8 in 2x400G mode by running the following command:
   ```
   sudo mlxconfig -d <device> -y s NUM_OF_PLANES_P1=0 MODULE_SPLIT_M0[0..3]=1
   MODULE_SPLIT_M0[4..7]=2 MODULE_SPLIT_M0[8..15]=FF NUM_OF_PF=2
   ```

5. If steps 3 or 4 were completed, power cycle the system for the configuration changes to take effect.
6. Disable Access Control Services (ACS)
   ACS must be disabled on the ConnectX-8 for the gpudirect test to work properly. To disable ACS, a script, disable_acs.sh, is provided in the partner diag package to assist in disabling ACS on the system under test.
   To disable ACS using the provided script, run the following command:
   ```
   sudo ./disable_acs.sh
   ```

4.2.20 dpudiag

- Purpose: Performs various NVIDIA network device diagnostic tests.
- To run this text, DOCA-OFED, numactl, and sshpass must be installed on the host system.

  If a Bluefield device is being tested, we recommend that you update to the latest available DPU OS version. Refer to Software Dependencies for more information.
- Loopback cables must be connected between ports of the devices under test. Refer to Required Hardware Configuration for more information.
- The configurable fields are listed in Table 4-18.

Table 4-18. dpudiag

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the timeout_sec value.
duration	Time in seconds	Integer	Specifies the test length.
system_prep	True/False	Boolean	Set up the prep for `dpudiag`, which must be run once per power cycle.
system_info	True/False	Boolean	Collects system information.
not_safe_mode	True/False	Boolean	Use this option when the host server root filesystem resides on an NVME drive.
debug	True/False	Boolean	Turns the verbose output on and off.
players	Array of Bluefield or ConnectX device descriptions. Bluefield-3 devices should be placed under 'bf' and ConnectX should be placed under "cx". Tests that run traffic between two single port devices require that a partner device is specified.	Array of json	Defines the devices to test.
ip	Bluefield-3 DPU IP address of oob_net1 interface	String	Specifies the IP address of the device under test. This field is only used when testing Bluefield-3 devices.
password	Bluefield-3 DPU Password	String	Specifies the password of the device under test.

Argument	Valid values	Type	Purpose
			This field is only used when testing Bluefield-3 devices.
pci	PCI address in format of Domain:Bus:Device	String	Specifies the PCI address of the device under test.
accelerators	• Bluefield: ib_traffic, mlxlink_counter, pcie_eye, memtester, burncortex, stress_ng, maxpwr_cpu, stress_ng_intense • ConnectX: ib_traffic, mlx_counter, pcie_eye	Array of strings	Specifies the tests to be run. Leaving this field empty will run the tests that are applicable for the device.

Example Spec:

```
{
    "virtual_id" : "NetworkInterfaceTraffic",
    "action" : "dpudiag",
    "args" : {
        "timeout_sec" : 4500,
        "config" : {
            "general": {
                "duration": 4200,
                "setup_prep":false,
                "system_info": true,
                "not_safe_mode": false,
                "port_ip_subnet": null,
                "disabled_measurements": [],
                "debug": false
            },
            "players": [
                {
                    "bf": {
                        "ip": "10.10.10.10",
                        "password": "<password>",
                        "pci": "0006:03:00",
                        "sd_pci": null
                    },
                    "cores": null,
                    "accelerators": [ "ib_traffic", "mlxlink_counter" ]
                },
                {
                    "bf": {
                        "ip": "10.11.10.11",
                        "password": "<password>",
                        "pci": "0016:03:00.0",
                        "sd_pci": null
```

```
                    },
                     "partner": {
                            "bf": {
                                   "ip": "10.11.10.11",
                                   "password": "<password>",
                                   "pci": "17:03:00.0”,
                                   "sd_pci": null
                               },
                               "accelerators": [
                                   "ib_traffic",
                                   "mlxlink_counter"
                                ]
                    "cores": null,
                    "accelerators": [ "ib_traffic", "mlxlink_counter" ]
                },
                {
                    "cx": {
                        "pci": "0016:03:00",
                        "sd_pci": null
                    },
                    "cores": null,
                    "accelerators": [ "ib_traffic", "mlxlink_counter" ]
                }
            ]
        }
    }
}
```

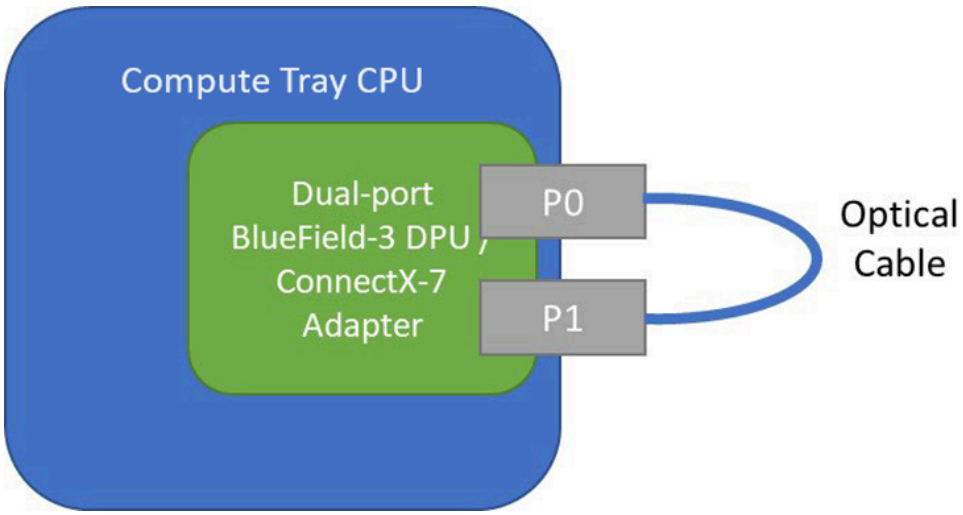
4.2.20.1 Required Hardware Configuration

The Bluefield-3 DPU and ConnectX NIC devices must be in loopback configuration (Port 0 and Port 1 connected) with an optical cable. Refer to Figure 4-20 for more information.

When selecting an optical cable, the cable must have the highest port rate designated for the dual-port BlueField-3 Networking Platform or ConnectX adapter, or alternatively, the rate intended for usage. For example, if a 2x200Gb/s BlueField-3 Networking Platform is being used, you need to use a 200Gb/s optical cable.

It is also important to opt for an optical transceiver that can use the maximum available power. For instance, when the intended usage scenario involves long-reach communication, a long-reach transceiver should be selected instead of a short-reach transceiver.

Figure 4-20. Loopback Cabling for BlueField-3 and ConnectX



When running tests on a BlueField-3 device, users must also connect the BF3 Arm cores to the same network as the host using the RJ45 port on the BF3.

4.2.20.2 Software Dependencies

This section provides information about the software dependences and instructions to install the dependencies.

1. Install DOCA libraries on the host.
   a. Go to https://developer.nvidia.com/doca-downloads.
   b. Select **Host-Server > DOCA-Host > Linux > arm64-sbsa > doca-all.**
   c. Select the target OS distribution and version.
   d. Follow the installation instructions for the preferred installer type.
2. Install the following dependencies and necessary tools.
   - sshpass
   - numactl
   - ipmitool
   - minicom
   - libibverbs1

   Here is an example of an installation on Ubuntu:

```
sudo apt update
sudo apt install -y doca-ofed sshpass numactl ipmitool minicom libibverbs1
```

3. If you are running on Bluefield-3, install the latest BFB version.
   a. Go to https://developer.nvidia.com/doca-downloads.
   b. Select **BlueField > BF-Bundle > Ubuntu > 22.04 > BFB**
   c. Follow the installation instructions for the BFB image.
4. Verify that the RShim service is available for each BF3.

```
ls /dev/rshim0/
ls /dev/rshim<1, 2, ...>/
```

   Each BF3 will have an rshim interface that starts at rshim0 and increments.

If rshim is not available on at least one device, to manually enable rshim, run the following command:

```
(sudo systemctl enable rshim);(sudo systemctl start rshim)
```

5. If you are running on a BF3, enable the BF3 DPU mode.

```
mlxconfig -d <mst_device> -y s INTERNAL_CPU_MODEL=1
```

6. Set the BF3 or CX7 to the operating mode you want.
   The dpudiag test supports the ethernet and infiniband modes. Refer to Using mlxconfig to set IB/ETH Parameters for more information about configuring ports.

7. AC power cycle the system for the software and configuration modifications to take effect.

4.2.21 powersync

- Purpose: Performs a synchronous CPU and GPU power stress.
- The configurable fields are listed in Table 4-20.

Table 4-20. powersync

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
tegra_telemetry	True/False	Boolean	Logs the CPU telemetry.
tegra_telemetry_time_ms	Time in milliseconds	Integer	Sets the polling rate for tegra_telemetry.
spec	thermalTestSpec_steadystate thermalTestSpec_cpugpupowerpulse	String	Controls the workload used for power stress. • `thermalTestSpec_steadystate` is for steady state CPU and GPU stress. • `thermalTestSpec_cpugpupowerpulse` is for synchronous pulsing of CPU and GPU.
cpu_gpu_sync_pulse_power_freqs	A comma-separated list of frequencies in Hz	String	Sets the frequencies to run during the synchronous CPU and GPU power pulsing. Must have the same number of entries as cpu_gpu_sync_pulse_power_duty_pcts and cpu_gpu_sync_pulse_power_runtimes_ms
cpu_gpu_sync_pulse_power_duty_pcts	A comma-separated list of duty cycles in percent	String	Sets the duty cycles to run during the synchronous CPU and GPU power pulsing. Must have the

Argument	Valid values	Type	Purpose
			same number of entries as cpu_gpu_sync_pulse_power_freqs and cpu_gpu_sync_pulse_power_runtimes_ms
cpu_gpu_sync_pulse_power_runtimes_ms	A comma separated list of runtimes in milliseconds	String	Sets the test duration. This value must always be smaller than timeout_sec. Must have the same number of entries as cpu_gpu_sync_pulse_power_freqs and cpu_gpu_sync_pulse_power_runtimes_ms

Example:

```
     {
           "virtual_id": "CpuGpuSyncPulsePower",
           "args": {
               "args": [
                   "spec=thermalTestSpec_cpugpupowerpulse",
                   "nvlink_mask=0x3FFFF",
                   "cpu_gpu_sync_pulse_power_freqs=3,10,100,500,1000,2000,4000,5000",
                   "cpu_gpu_sync_pulse_power_duty_pcts=50,50,50,50,50,50,50,50",

"cpu_gpu_sync_pulse_power_runtimes_ms=900000,900000,900000,900000,900000,900000,900000
,900000",
               ],
               "timeout_sec": 3600,
               "tegra_telemetry": true,
               "tegra_telemetry_time_ms": 300,
               "enable_prometheus": true
           },
           "action": "powersync"
       },
```

4.2.22 cable_cartridge

- Purpose: Checks the cable cartridge FRU slot information is programmed correctly.

  This test will perform checks from both the compute trays and switch trays.
- The configurable fields are listed in Table 4-16.

Table 4-21. cable_cartridge

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the `timeout_sec` value.
ishttps	true/false	Boolean	Selects if the redfish URI is http (false) or https (true).

Argument	Valid values	Type	Purpose
use_redfish_for_compute	true/false	Boolean	Chooses to use redfish (true) or ipmitool (false) for checking the cable cartridge FRUs.The user must provide the CBC EEPROM redfish URIs in the rf_endpoints field of the global_args if use_redfish_for_compute is true. Refer to Table 4-3 for more information.
fru_desc_regex_list	regex expression	List of strings	Used only when `use_redfish_for_compute` is false. Provides a list of regex expressions to match the FRU device description in ipmitool.
curl_options	curl command options and arguments	String	Provides additional options and arguments to the curl command when using https.

Example:

```
{
   "virtual_id": "CableCartridgeEepromCheck",
   "args": {
        "timeout_sec": 300,
         "fields_to_exclude_check":["board_serial","product_serial","board_date"],
          "use_redfish_for_compute" : true,
          “fru_desc_regex_list”:[“^CBC_.*”]
          "redfish_api_args" :
                {
                    "ishttps" : true,
                    "curl_options" : "--silent --insecure --user root:0penBmc"
                }
           },
    "action": "cable_cartridge"
}
```

4.2.24 environmentcheck

- Purpose: Checks that the right permissions and tools are set up before running the selected actions.
  - Each `virtual_id` in the spec file includes an *“env_check”* argument that lists the specific dependency checks it needs.
  - Some dependencies apply to all `virtual_id` in the spec file so they’re listed under the *“env_check”* list section of the *“environmentcheck”* action itself.
- The configurable fields are listed in Table 4-22.

Table 4-22 environmentcheck

Argument	Valid values	Type	Purpose
env_check	numa_allocation, interrupts, tegra_mods, consistent_vbios, gpu_pcie_presence, gpu_bus_connectivity	string	Determines which environment check to perform.

Example:

```
    {
        "virtual_id": "environmentcheck",
        "env_check": ["gpu_pcie_presence", "gpu_bus_connectivity","consistent_vbios"],
        "action": "environmentcheck"
     }
```

4.2.25 tegra_cpu TegraCper

- Purpose: Collects and decodes the CPER errors stored in the Grace R/W SPI flash on the system under test.
- The TegraCper test is informational, and will fail only if the provided ruleset file is invalid or the tool is unable to read the CPERs from the R/W SPI flash.
- This test is not included in mfg or field by default. Users will need to add it to the spec json to run the test.
- The configurable fields are listed in Table 4-5.

Table 4-23. tegra_cpu TegraCper

Argument	Valid values	Type	Purpose
timeout_sec	Time in seconds	Integer	Kills the test if the test time exceeds the timeout_sec.

Example spec:

```
{
   "virtual_id": "TegraCper",
   "args": {
      "timeout_sec": 600,
      "args": [
         "-prod", "GB200-NVL 2:4 board PC",
         "-test", "19"
            ]
      },
   "action": "tegra_cpu"
}
```

4.2.25.1 Software Dependencies

This section provides information about the software dependences for the TegraCper test and instructions to install the dependencies.

1. The Linux Kernel version must be 6.8.1016 or later.
2. The installed SBIOS version must be 02.03.00 or later.
3. The user must build and install the cper-dump kernel module prior to TegraCper

test execution.

a. To install the cper-dump kernel module, download the repository, available at https://github.com/NVIDIA/cper-dump
b. Build and insert the kernel module using the following commands:

```
tar xvf cper_dump.tar.gz
cd cper_dump
make clean
make
sudo insmod ./cper_dump.ko
```

4.3 SKU JSON File

The SKU file contains information about the inventory items, including the firmware versions, FRU information, and so on.

The inventory test compares the SKU file content with the content on the system. When a new item is added to the system, it must be added to the SKU file to be checked in the inventory test. Figure 4-3 is an example of a SKU JSON file, and the object that will be added is a GPU, which is identified by its BDF at 0009:01:00:0. Each of its characteristics is entered as an item in `attributes`.

By default, the VBIOS check is configured to compare the VBIOS version with the regex `.*`. Customers can modify the **`VBIOS_version`** field to perform a VBIOS version check. If `matchtype` is set to `"exact"`, the inventory test will check for an exact match. If `matchtype` is set to `"min"`, the inventory test will check for versions greater than or equal to the value.

Figure 4-3. SKU JSON File

```
{
    "filter": "",
    "type": "GPU",
    "id": "0009_01_00_0",
    "description": "",
    "attributes": [
        {
            "key": "VendorID",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "10de",
            "validateRegex": ".*"
        },
        {
            "key": "DeviceID",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "237e",
            "validateRegex": ".*"
        },
        {
            "key": "SSVendorID",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "10de",
            "validateRegex": ".*"
        },
        {
            "key": "SSDeviceID",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "1809",
            "validateRegex": ".*"
        },
        {
            "key": "PCIWidth",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "x1",
            "validateRegex": ".*"
        },
        {
            "key": "VBIOS_version",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "96.00.62.00.01",
            "validateRegex": ".*"
        },
        {
            "key": "InfoROM_version",
            "valuetype": "string",
            "matchtype": "exact",
            "value": "G530.0206.01.01",
            "validateRegex": ".*"
        }
    ]
}
```

Chapter 5. Field Diagnostics Guidelines

Field diagnostics are used to identify and isolate faults in NVIDIA hardware and determine if an NVIDIA part is eligible for RMA. Before running the field diagnostics, users should ensure that they have completed the prerequisites in Executing Partner Diagnostics on the System.

Here are the files that are required to run the field diagnostics:

- `partnerdiag`
- `onediagfield.rXX.YY`
- `Test Spec JSONs:`
  - `spec_<product-name>_field_level1.json`
  - `spec_<product-name>_field_level2.json`
  - `spec_<product-name>_field_ist.json`
- `SKU JSON: sku_<product-name>_field.json`
- `fdmain.sh`

Caution: Users might need to modify the **`BaseboardsPciIds`** field of the field spec json files to match their system's GPU PCI addresses. Refer to Table 4-3 for information about modifying the **`BaseboardsPciIds`** field.

Users should modify **only** the **`BaseboardsPciIds`** field in the field diagnostics test spec or the SKU JSON files unless directed by an NVIDIA representative.

Here is an example command that runs the field diagnostics:

```
sudo ./partnerdiag --field –run_on_error --no_bmc
```

Note: When running GPU field diagnostics on GB200 with CX8 or GB300, users must add the --gpu_fd_pex_link_auto argument.

L11 field rack diagnostics can run in the following modes:

- Self-Hosted, where the diag that users run on the primary node will copy the diag to the secondary nodes and execute them.
- Managed, where users need to manually copy the diag into the primary and the secondary nodes and execute them.

Note Managed mode is not supported in the L11 partner diag packages for v0.9.09

Caution: In the Managed mode, when you use one of the compute trays as the primary node, to test that compute node, copy the diagnostics into a folder for the primary diag and another folder for the secondary diag.

Run the primary and secondary diag commands separately on these folders.

Unix Sockets for the Managed Mode

The L11 rack diagnostics support using TCP/IP stack or Unix sockets for managed mode and using Unix sockets requires additional setup.

Note: The setup steps for when running the primary diagnostics from one of the compute trays of the system under test differ from the steps for when running from a separate management node.

To set Unix sockets when running the primary diagnostics from the SUT (System Under Test):

1. Log in to the primary compute node.
2. Run the following command to generate the SSH keys.

   ```
   ssh-keygen -t rsa -b 4096
   ```
3. Run the following command in the folder that contains the primary diagnostics.

   ```
   sudo ./partnerdiag --field --managed_mode
   --num_nodes_to_connect=<total_number_of_compute_nodes> --gdm_fd=@computenode0
   ```
4. Run the following command in the folder that contains the secondary diagnostics.

Note: In the spec file, the onediag_gdm_fd for the system being used as the primary should be the same as the gdm_fd specified on the command line.

```
sudo ./partnerdiag --field --managed_mode --gdm_fd=@computenode0
--dra_client_fd=@draserverfd
```

5. Run the following commands on the primary compute node for each of the remaining compute nodes that are not the primary compute nodes.

   ```
   ssh-copy-id username@<ipaddress_of_compute_node>


   ssh -t username@<ipaddress_of_compute_node> 'killall socat'


   socat "ABSTRACT-CONNECT:computenode0" SYSTEM:"ssh
   username@<ipaddress_of_compute_node> -- 'socat STDIO
   ABSTRACT-LISTEN:computenode<1...n-1>,reuseaddr,fork'" &


   socat "ABSTRACT-CONNECT:computenode0" SYSTEM:"ssh
   username@<ipaddress_of_compute_node> -- 'socat STDIO
   ABSTRACT-LISTEN:computenode<1...n-1>_onediag,reuseaddr,fork'" &


   socat "ABSTRACT-CONNECT:draserverfd" SYSTEM:"ssh
   username@<ipaddress_of_compute_node> -- 'socat STDIO
   ABSTRACT-LISTEN:draclientfd<1...n-1>,reuseaddr,fork'" &
   ```
6. Run the following command on all of the secondary compute nodes.

   ```
   sudo ./partnerdiag --field --managed_mode  --dra_client_fd=@draclientfd<1…n-1>
   --gdm_fd=@computenode<1…n-1>
   ```

   Where *<1…n-1>* is the index of the compute node on which the command is being executed starting from 1 and assuming the primary compute node is index 0.

To set up Unix sockets when running the primary diagnostics from a separate management node:

1. On the primary management node, run the following command to generate the SSH keys.

```
ssh-keygen -t rsa -b 4096
```

2. Run the following command on the to launch the partner diagnostics on the primary management node.

```
sudo ./partnerdiag --field --managed_mode
--num_nodes_to_connect=<total_number_of_compute_nodes> --gdm_fd=@managementnode
```

3. Run the following commands on the primary management node for each of the compute nodes.

```
ssh-copy-id username@<ipaddress_of_compute_node>


ssh -t username@<ipaddress_of_compute_node> 'killall socat'
socat "ABSTRACT-CONNECT:managementnode" SYSTEM:"ssh
username@<ipaddress_of_compute_node> -- 'socat STDIO
ABSTRACT-LISTEN:computenode<0...n-1>,reuseaddr,fork'" &


socat "ABSTRACT-CONNECT:managementnode" SYSTEM:"ssh
username@<ipaddress_of_compute_node> -- 'socat STDIO
ABSTRACT-LISTEN:computenode<0...n-1>_onediag,reuseaddr,fork'" &


socat "ABSTRACT-CONNECT:draserverfd" SYSTEM:"ssh
username@<ipaddress_of_compute_node> -- 'socat STDIO
ABSTRACT-LISTEN:draclientfd<0...n-1>,reuseaddr,fork'" &
```

4. Run the following command on each of the compute nodes.

```
sudo ./partnerdiag --field --managed_mode --dra_client_fd=@draclientfd<0...n-1>
--gdm_fd=@computenode<0...n-1>
```

Where *<0…n-1>* is the index of the compute node on which the command is being executed starting from 0.

Here are example commands that run the field diagnostics for the L11 rack:

Self-Hosted Mode

```
sudo ./partnerdiag --field --primary_diag_ip=<IP>
```

Managed Mode Primary Node

```
sudo ./partnerdiag --field --managed_mode --num_nodes_to_connect=<NUM>
[--primary_diag_ip=<IP> | --gdm_fd=<@socketfd>] [--global_dra_server_port=<PORT>]
```

Managed Mode Secondary Node

```
sudo ./partnerdiag --field --managed_mode [--primary_diag_ip=<IP> |
--gdm_fd=<@socketfd>] [--global_dra_server_port=<PORT> | --dra_client_fd=<@socketfd>]
```

Caution: In the Managed mode, start the diag on the primary node first, wait for 60 secs and then start the diag on all the secondary nodes. Primary diag needs all the secondary

	diags to be connected to primary in 300 secs after it starts.

Note: In the L10 compute tray partner diagnostics --field, the default configuration uses the spec json files for GB200 designs where the GPU is enumerated at gen 4.

- For designs where the GPU is enumerated at gen 5, users should override the default field spec json file by running the following command:`--run_spec=spec_gb200_nvl_2_4_board_pc_field_level2_cx8_qs.json`
- For designs where the GPU is enumerated at gen 6, users should override the default field spec json file by running the following command:`--run_spec=spec_gb200_nvl_2_4_board_pc_field_level2_cx8.json`

Table 5-1 provides a list of some helpful arguments.

Table 5-1. Helpful Arguments for Field Diagnostics

Option	Description
--level1	Runs the level 1 tests of the field diagnostics.
--level2	Runs the level 2 tests of the field diagnostics.
--gpufielddiag	Runs the GPU Field Diagnostics.
--arm_ffa_src=<path>	Provides the path to the `arm_ffa` source directory. The path is typically `<linux_source_tree>/drivers/firmware/arm_ffa`. If the host OS does not natively have the `arm_ffa` module, for the diagnostic package to correctly function, this argument must be used for the diagnostic package.
--log <absolute_path>	The path where the logs of this OneDiag run will be generated.
--run_on_error	Runs the specified tests and does not stop on failures.
--no_bmc	Runs without any BMC-related tasks. The BMC retrieves information about the system configuration and sensor data. This option disables correct product identification and prevents sensor monitoring and incorrect firmware version detection.
--fail_on_first_error	Runs all tests and fails on the first failure.
--help	Prints a complete list of arguments for manufacturing and field modes.
--skip_tests=<virtual ID>	Skips the test with the virtual ID that is provided as an argument to this option. Multiple tests can be skipped if the tests are provided as a comma-separated list of virtual IDs without spaces.
--unskip_tests=<virtual ID>	Unskips tests with the virtual ID provided as an argument to this option where the give virtual ID has skip_test set to true in the test json. Multiple tests can be unskipped if they are provided as a comma-separated list of virtual IDs without spaces.
--test=<virtual ID>	Runs only the test with the virtual ID that is provided as an argument to

Option	Description
	this option. Multiple tests can be run if they are provided as a comma-separated list of virtual IDs without spaces.
--primary_diag_ip=<Primary Diag IP>	Specifies the IP address of the primary node from which the diagnostic is being executed. This argument is only available in the partner diagnostics for L11 rack.
--global_dra_server_port=<port>	Port on which the DRA listens (primary node) or to which the DRA connects (secondary node).
--gdm_fd=<socket>	Unix/Abstract socket fd where the GDM listens (primary node) or the GDM connects to (secondary node). • Used in the managed mode. • For abstract sockets, ensure that you use @ at the beginning.
--num_nodes_to_connect=<number>	Total number of secondary nodes that need to connect to the primary and is while running in the managed mode in the primary node.
--dra_client_fd=<socket>	Unix/Abstract socket fd to which the DRA client that is running on the secondary node is connected.
--ist	Runs IST Diagnostic tests. Refer to IST Field Diagnostics Guidelines for more information.
--gpu_fd_pex_link_auto	Don't force the PCIe gen speed and link width for GPU field diagnostics. This argument is required when running GPU field diagnostics on GB200 with CX8 or GB300.
--partner_extra_logging=<logs>	Runs the partner diagnostic with power, thermal, voltage or clock logged. Logging interval can be modified using --partner_extra_logging_ms Example usage: --partner_extra_logging=thermal,clock
--partner_extra_logging_ms	Sets the logging interval in milliseconds of the telemetry logged in --extra_partner_logging. The default value is 500ms and minimum value is 100ms.

Level 1 and Level 2 have equivalent coverage and test time for this release of field diagnostics for the compute tray.

Table 5-2. Compute Tray Field Diagnostics Test Modes and Approximate Completion Times for GB200

Test	Test Duration	Level 1	Level 2	Test Description
Checkinforom	40 seconds	Yes	Yes	Checks the inforom
Inventory	Approximately 1.5 minutes.	Yes	Yes	System-level check of components against the expected versions.
TegraCpu	Approximately 31 minutes.	Yes	Yes	Performs CPU diagnostics testing.
TegraCpu4	Approximately 30 minutes	Yes	Yes	Performs CPU diagnostics testing.
TegraCpu5	Approximately six minutes	Yes	Yes	Performs CPU diagnostics testing.

Test	Test Duration	Level 1	Level 2	Test Description
TegraMemory	Approximately one minute.	Yes	Yes	Concurrent data traffic generated by High Speed Scrubbing. This requires a MODS secure partition.
CpuMemorySweep	Approximately one hour.	Yes	Yes	Perform reads and writes and correctness checks for CPU memory. This requires a MODS secure partition.
TegraClink	Approximately 10 minutes.	Yes	Yes	CPU-CPU NVLink bandwidth, eye diagram tests.
Gpustress	Approximately four minutes.	Yes	Yes	GPU Stress Tests.
Gpumem	Approximately one minute.	Yes	Yes	GPU memory and interface tests.
Pcie	Approximately six minutes.	Yes	Yes	PCIE Bandwidth, speed switching, and eye diagram tests.
C2C	Approximately two minutes.	Yes	Yes	CPU-GPU C2C NVLink stress.
CPUVDD_PowerStress	Approximately 30 minutes.	Yes	Yes	Performs high-power stress on the CPUs.
ThermalSteadyState	Approximately 15 minutes.	Yes	Yes	Stress power on system components (CPU and GPU).
Connectivity	Approximately 12 minutes.	Yes	Yes	Validates the electrical quality of NVLinks and PCIE link speeds/width match the POR.
NvlBwStress	Approximately 12 minutes.	Yes	Yes	GPU-GPU NVLink bandwidth and eye diagram tests.
NvlBwStressBg610	Approximately 12 minutes	Yes	Yes	GPU-GPU NVLink bandwidth stress with background steady state GPU stress.
CpuGpuSyncPulsePower	Approximately 20 minutes (2.5 minutes per frequency)	Yes	Yes	Synchronous CPU and GPU pulsing power stress.
DimmStress	Approximately six minutes.	Yes	Yes	Stresses CPU DRAM.
gpufielddiag	Approximately 2.25 hours	N/A	N/A	GPU field diagnostics test suite.

Table 5-3. L11 Rack Field Diagnostics Test Modes and Approximate Completion Times for GB200

Test	Test Duration	Test Description
Connectivity	Approximately 16 minutes.	Validates that the electrical quality of NVLinks meets the POR.
NvlBwStress	Approximately 21 minutes.	Stresses NVLink traffic.
NvlBwStressBg610	Approximately 21 minutes	Stresses NvLink traffic but more computationally stressful than NvlBwStress as there is a GPU compute test running in the background
NvlBwStressBg610Pulsy	Approximately 21 minutes	Similar to NvlBwStressBg610 but the background workload is pulsing
CpuGpuSyncPulsePower	Approximately 23 minutes.	Performs a synchronous CPU and GPU pulsing workload for all compute trays in the L11 rack.
ThermalSteadyState	Approximately 22 minutes.	Perform reads and writes and correctness checks for CPU memory. This requires a MODS secure partition.

5.1 IST Field Diagnostics Guidelines

IST Field Diagnostics is an enhancement to the existing NVIDIA Partner Diagnostics package. The initial version of IST field diagnostics are available for GB200 after the v0.9.00 full software release.

5.1.1 Launching IST Field Diagnostics

Before you run IST Field Diagnostics, you need to configure it. .

- To manually enable/disable the IST OOB PRC knob, refer to IST Setup: Redfish Commands.
- To automatically enable/disable the IST OOB PRC knob by using the IST Field Diagnostic Tool, refer to IST Setup: Test Spec File.

Note The IST Field Diagnostics test requires a separate IST image to run. Due to the size of the image, it is distributed separately from the main Partner Diagnostics package.

To download the IST image for the GB200, go to the following NVOnline IDs:

- For GB200 v0.9.00 Partner Diagnostics, go to NVOnline: **1128244**.
- For GB200 v1.0 or later Partner Diagnostics, go to NVOnline: **1130090.**
- For GB300 Partner Diagnostics, go to NVOnline: **1142068.**

Caution: For the diag to run correctly, the IST image folder **must** be in the `spec_<product_name>_field_ist.json` directory.

Here are examples for running IST Field Diagnostics:

Using IST Diagnostic Tool To Enable/Disable IST PRC Knob

```
sudo ./partnerdiag --field --ist --run_on_error
```

To configure the IST test spec file to run Redfish commands and to enable and disable the IST OOB PRC knob, refer to IST Setup: Test Spec File for more information.

Since there are multiple partnerdiag releases for specific firmware versions, to launch IST test cases, run one of the following commands

- To run IST with the GB200 v0.9.00 diagnostics and firmware bundle, run the following commands:

  ```
  sudo ./partnerdiag --field --ist
  --run_spec=spec_<board_type>_field_ist_fw_bundle_09.json
  ```
- To run the IST with the GB200 v1.1 or later diagnostics and firmware bundle, run the following commands:

  ```
  sudo ./partnerdiag --field --ist
  --run_spec=spec_<board_type>_field_ist_fw_bundle_10.json
  ```

Without Using IST Diagnostic Tool To Enable/Disable IST PRC knob

```
sudo ./partnerdiag --field --ist --run_on_error --no_bmc
```

to enable and disable the IST OOB PRC knob without using the IST Diagnostic tool, refer to IST Setup: Redfish Commands for more information.

Before you start the IST tests, ensure the test spec is correctly set. There will be various progress messages and a PASS/FAIL banner after the test has completed.

Caution: Do not release a system back into production without a clean **PASS** from the IST Field Diagnostic. If the GPU PRC knobs are not successfully locked by the IST Field Diagnostic, customers can place their GPUs in the IST non-responsive state using an in-band command.

Restore The GPUs

If customers do not receive the PASS banner from the previous test, to recover the GPUs, run the following command. A pass banner for the following command indicates that the GPU PRC knobs are successfully locked.

```
sudo ./partnerdiag --field --ist --test=RestoreGpu
```

There is a known issue where the `RestoreGpu` command might fail. If this occurs, the failure signature will resemble the following:

```
              | OK
DGX-000000000000  | RestoreGpu          | ist            | GPU0 failed to be controlled by HMC after retrying 18 times | GPU       |
redfish/v1/Managers/HGX_BMC_0/Actions/Oem/NvidiaManager.NSMRawCommand | OK
DGX-000000000000  | RestoreGpu          | ist            | GPU1 failed to be controlled by HMC after retrying 18 times | GPU       |
redfish/v1/Managers/HGX_BMC_0/Actions/Oem/NvidiaManager.NSMRawCommand | OK
DGX-000000000008  | RestoreGpu          | ist            | gpu_ist                                                     | GPU       | GPU IST
              | OOB redfish command checkHMCGPUStatus failed on all devices. Verify testargs have been configured correctly
DGX-000000000000  | AutomaticRepair     | repair_utility | Test Post Processing                                        | GPU       |
              | OK - Nothing to repair
```

Perform a cold reboot of the test machine and run the `RestoreGpu` command again.

5.1.2 IST Test Categories

The existing IST Field Diagnostics Tool executes all tests at one time. There are plans to introduce three additional options, which will allow users to run logic, SRAM, or DRAM tests individually. Currently there is a command-line option, but the logic to enable the test selection has not yet been added, so all of the four test categories in Table 5-4 run the same IST tests.

Table 5-4. IST Test Categories

Type	Arguments	Description
Logic test	--logic	Runs IST tests for logical gates, including computing units and memory controllers. `sudo ./partnerdiag --field --ist --logic`
SRAM test	--sram	Runs MBIST (Memory Built-In Self Test) on SRAM. `sudo ./partnerdiag --field --ist --sram`
DRAM test	--dram	Runs PMBIST (Programmable Built-In Self Test) on HBM. `sudo ./partnerdiag --field --ist --dram`
All test	N/A	No argument is required as the tool will run all categories ie. Logic, SRAM, and DRAM tests by default. `sudo ./partnerdiag --field --ist`

5.1.3 IST Setup: Redfish Commands

Before you run IST tests, enable the IST oneshot mode. After completing the IST tests, you **must** disable the IST oneshot mode. Customers **must** repeat the following commands for all the GPUs.

Enable the IST OOB PRC Knob

```
curl -i -k -u $BMC_USER:$BMC_PASS
https://<bmc_ip>/redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_<gpu_id> -X PATCH
-H "Content-Type: application/json" -d '{"Oem": {"Nvidia":
{"InbandReconfigPermissions":{"InSystemTest":{"AllowOneShotConfig": true}}}}}'
```

Disable the IST OOB PRC Knob

```
curl -i -k -u $BMC_USER:$BMC_PASS
https://<bmc_ip>/redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_<gpu
_id> -X PATCH -H "Content-Type: application/json" -d '{"Oem": {"Nvidia":
{"InbandReconfigPermissions":{"InSystemTest":{"AllowOneShotConfig": false}}}}}'
```

Verify the IST OOB PRC Knob

```
curl -i -k -u $BMC_USER:$BMC_PASS
https://<bmc_ip>/redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_<gpu
_id>
```

Use the command to verify the knob to get the value of `AllowOneShotConfig` as shown below.

- If users are trying to enable the IST OOB PRC knob, the value will be true.
- If users are trying to disable the knob, the value will be false

```
{
    ...
  "Oem": {
              "Nvidia": {
              ...
              "InbandReconfigPermissions": {
              ...
              "InSystemTest": {
                "AllowFLRPersistentConfig": false,
                "AllowOneShotConfig": false,
                "AllowPersistentConfig": false
              },
  ...
}
```

IST Test configuration spec change

To ensure that the IST diagnostic tool does not enable/disable the knob, modify the `spec_<product_name>_field_ist.json` IST test configuration file and set `set_oob_gpu_prc_knob` to false in the `Enable_IST` test and `restore_oob_gpu_prc_knob` to false in the `RestoreGpu` test.

```
{
            "actions": [
            {
            "virtual_id": "Enable_IST",
            "args": {
                "set_oob_gpu_prc_knob": false,
                "use_redfish_for_prc_knob": true,
                "continue_on_setup_error": true,
                "skip_ist_test": true,
                "timeout_sec": 2400
            },
            "osm": {
                "triggers": {
                    "DGX.000000000008": "EXIT",
                    ".*": "NEXT"
                }
            },
            "action": "ist"
            },

            {
            "virtual_id": "RestoreGpu",
            "action": "ist",
            "args": {
                "restore_oob_gpu_prc_knob": false,
                "use_redfish_for_prc_knob": true,
                 "restore_cc_gpu_prc_knob": true,
                "skip_ist_test": true,
                "timeout_sec": 800
            }
```

```
            }
            ],
            "version": "2"
}
```

5.1.4 IST Setup: Test Spec File

All IST test configurations are in the `spec_<product_name>_field_ist.json` file. IST Field Diagnostics Tool supports running Redfish commands to set up the IST OOB PRC knob. This guide provides information about configuring the Redfish raw commands in the test spec to fit a customers' system. Customers need to specify the Redfish raw commands for each GPU to enable (or disable) the IST OOB PRC knob.

To access the Redfish raw command fields, in the text file, go to **global_args** > **bmc** > **customCommands** > **enableIstPrcKnob/disableIstPrcKnob** > **redfish_raw_commands**.

- The Redfish raw commands listed under `enableIstPrcKnob` enable the IST OOB PRC knob.
- The commands listed under `disableIstPrcKnob` disable the IST OOB PRC knob.

Table 5-5. IST OOB PRC Knob Redfish Raw Command Fields in the Test Spec

Field	Type	Description
requests_util	string	Specifies a custom request method to communicate with the Redfish server. Valid methods include `GET`, `POST`, and `PATCH`. It is equal to the `-X` option in the curl command.
api	string	The Redfish command API is `redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_<idx>`, where *idx* represents the GPU number.
headers	object	Additional header to include when sending HTTP requests to the Redfish server. This is equal to the `-H` option in the curl command.
json	object	Sends the specified data, in the JSON format, in a POST/PATCH request to the Redfish server. This is equal to the `-d` option in the curl command.
expected_data	object	Checks whether the configuration is correctly set. The object key refers to the Redfish API for GET requests, with `keys` and `value` properties. `keys` represents the path in the GET response, and `value` indicates the expected value at that path. Multiple expected data can be specified.

Here is an example of a Redfish raw command for one GPU. It corresponds to the Redfish commands in Enable IST OOB PRC Knob.

```
{
    "requests_util": "PATCH",
    "api": "redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_1",
    "headers": {
        "Content-Type": "application/json"
    },
    "json": {
        "Oem": {
            "Nvidia": {
                "InbandReconfigPermissions":{
                    "InSystemTest":{
                        "AllowOneShotConfig": true
                    }
                }
            }
        }
    },
    "expected_data": {
        "redfish/v1/Systems/HGX_Baseboard_0/Processors/GPU_1": {
            "keys": ["Oem", "Nvidia", "InbandReconfigPermissions", "InSystemTest",
"AllowOneShotConfig"],
            "value": true
        }
    }
}
```

You also need to set up the BMC Redfish credentials for the IST Field Diagnostics Tool. To specify credentials in the test spec file, use the example under global_args.

```
     "bmc_redfish_credentials" :
       {
           "username" : "abc",
           "password" : "xyz"
       }
```

Caution IST tests were first enabled with the v0.9.00 Diag release, so these tests can only be run with the v0.9.00 or later firmware recipe.

IST Diag 1.0 can only be run with the v1.0 or later firmware bundles.

Chapter 6. Interpreting the Diagnostic Results

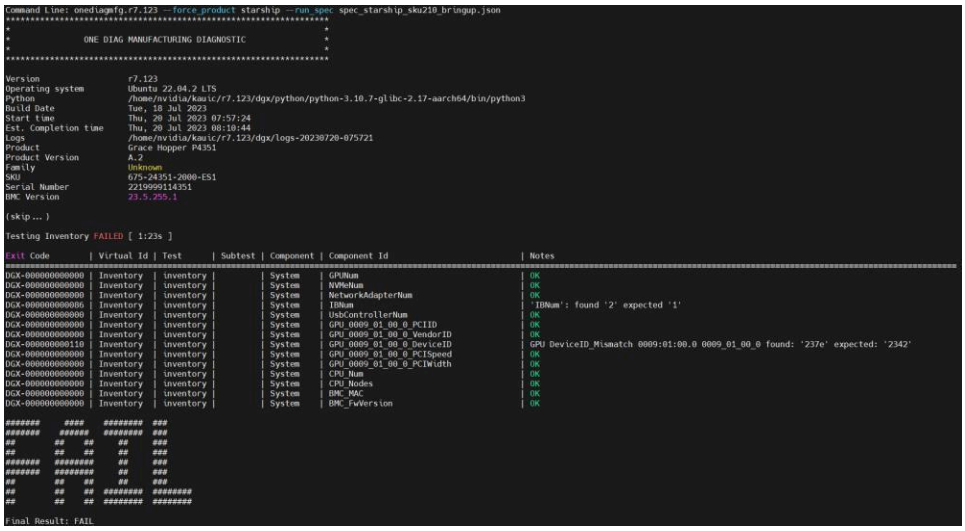
Refer to the *Debug and RAS Guide for NVIDIA Datacenter Products* (NVOnline: **1109712**) for diagnostics error code to action mapping.

Refer to the *String Format for Reporting Physical Location with Error Messages* (NVOnline: **1125990**) for information about decoding the **Notes** section when an NVLink error occurs.

6.1 PASS/FAIL/RETEST Banner

Here is an example of the FAIL banner on MGX products:

Figure 6-1. The FAIL Banner



From the output in Figure 6-1, you can see which test failed and for which component. The output also shows the log file location (refer to Retrieving Log Files for more information).

Figure 6-2 shows an example of the PASS banner.

Figure 6-2. The PASS Banner

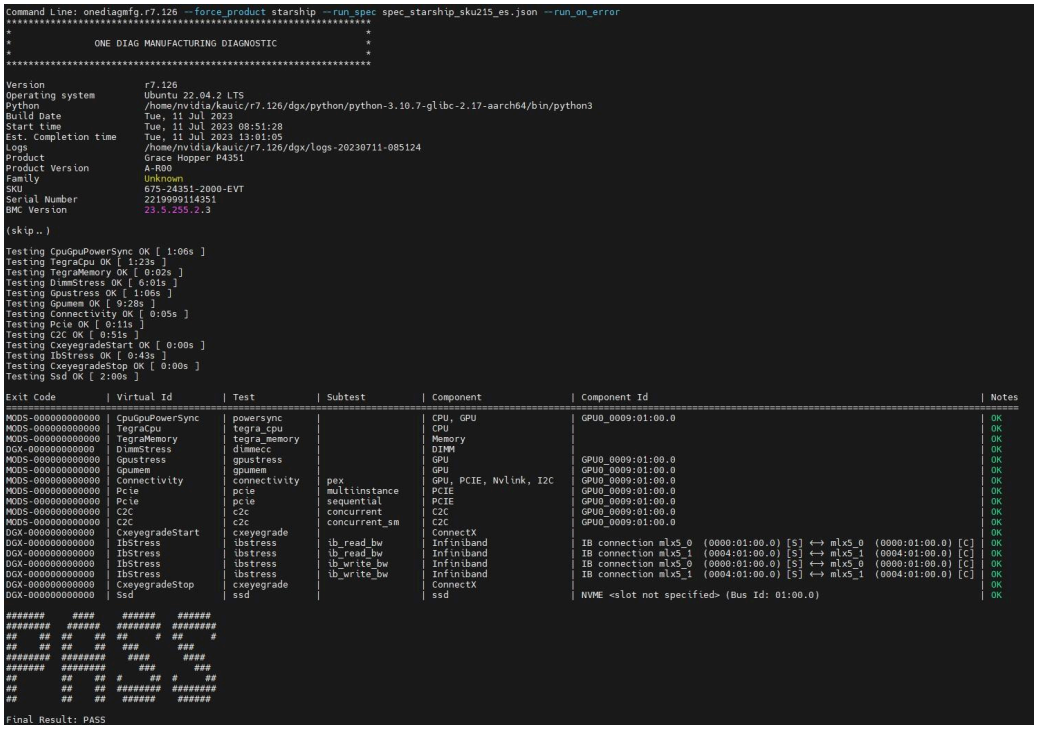
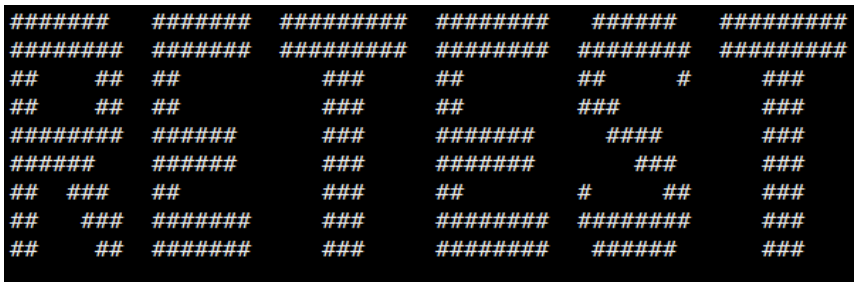
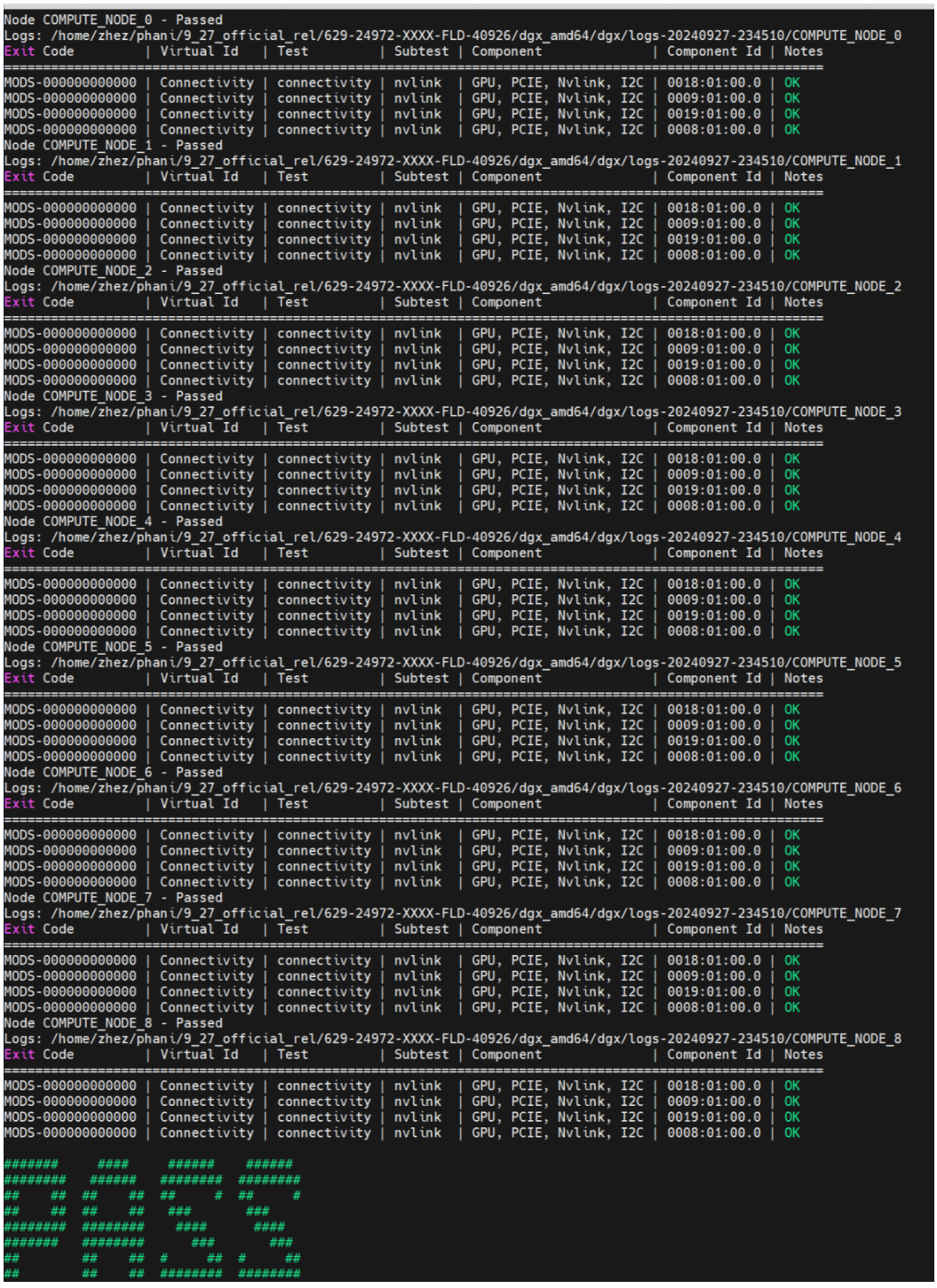


Figure 6-3. The RETEST Banner



The L11 Rack diagnostics output will contain test result tables for each compute or switch tray targeted during the test runtime. Figure 6-4 shows the banner from the L11 Rack diagnostics.

Figure 6-4. The L11 Rack Banner



6.2 Retrieving Log Files

This section provides information about how to retrieve log files.

6.2.1 Log File Organization

Partner Diagnostics generates logs in the `dgx/logs-<yyyymmdd-hhnnss>` folder.

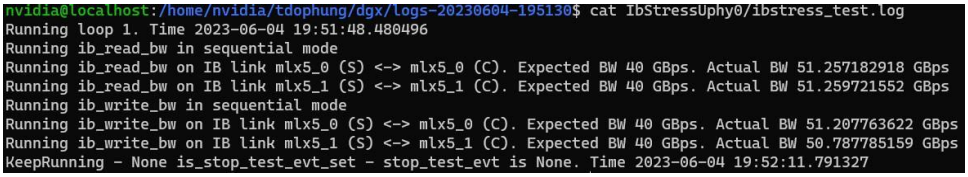
The log folder name is based on the test date and timestamp in the `logs/logs-yyyymmdd- hhnnss` format, where:

- `mm` = Month
- `dd` = Day
- `yy` = Year
- `hh` = Hours
- `nn` = Minutes
- `ss` = Seconds

6.2.2 Logs for Each Test

Logs for each test are saved in their corresponding test subfolders, and here is an example:

Figure 6-5. Example of a Test Log



Here are some notable logs from these subfolders:

- TegraCpu: `output.log`
- Gpustress/Gpumem/Pcie/Connectivity: in the `GPU<number>_<bdf>/output.log` file.

This log contains the GPU power and thermal telemetry data.

6.2.3 Logs for a General Run

In addition to logs for each test, Partner Diagnostic generates background processes to log additional global telemetry data, such as BMC SEL log output, Dmesg, and so on.

Table 6-1. General Run Logs

Log name	Description
bmcevents.log	BMC events.
output.log	The processes run on the command line when Partner Diagnostic is invoked.
run.log	Plain-text logs of the Partner Diagnostic run.

Log name	Description
Exception.log	Python exceptions that occurred but were missed and reported by the software. These exceptions might be due to software and hardware errors.

These logs will be in the `dgx/logs-<yyyymmdd-hhnnss>` log folder.

6.3 CSV Log File

The Partner Diagnostic generates a CSV log file (`summary.csv`) and stores it in the `logs-<yyyymmdd-hhnnss>` folder.

The files interpret the results for a specific run and contain the following information:

- **Exit code**: An exit code apart from 0 that indicates an error.
- The test and subtest name.
- **Component**: This is in the following format:
  `<GPU slot number> <PCI bus>_<PCI device>.<PCI function> (SN_< GPU serial number>) (<tray location: upper or lower>)`

  For example: `SXM2 3b_00.0 (SN_0332318503329) (lower tray)`
- **Notes**: Specifies the failure message.

6.4 JSON Log File

The Partner Diagnostic also generates a summary.json and stores it in the logs-<yyyymmdd-hhnnss> folder.

The JSON array stored in the file interprets the results for a specific run and contains test result specific objects with the following information:

- **Error Code**: An error code in the following format: `XXX-YYY-Z-<12 digit Error Code>`. A 12-digit error code that is not `000000000000` indicates an error.
- **Test**: The test name.
- **Virtual ID**: The test action name.
- **Component ID**: This is in the following format:

  `<GPU slot number> <PCI bus>_<PCI device>.<PCI function> (SN_< GPU serial number>) (<tray location: upper or lower>)`

  For example: `SXM2 3b_00.0 (SN_0332318503329) (lower tray)`
- **Notes**: Specifies the failure message.

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, the NVIDIA logo, and NVLink 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.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Arm

Arm, AMBA, and ARM Powered are registered trademarks of Arm Limited. Cortex, MPCore, and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. ″Arm″ is used to represent ARM Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS, and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com