UFM Cable Validation Tool v1.2.0

Exported on Nov/27/2023 05:28 PM
https://docs.nvidia.com/networking/x/Nw5GC

Table of Contents

About This Document

This document describes NVIDIA® Unified Fabric Manager (UFM®) Cable Validation tool, connectivity and configuration options.

Technical Support

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

- E-mail: enterprisesupport@nvidia.com
- Enterprise Support page: https://www.nvidia.com/en-us/support/enterprise

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

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

Document Revision History

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

Release Notes

Limitations

Internal Reference Number	Issues
N/A	**Description**: The tool does not support unmanaged switches
	Keywords: Unmanaged Switch
	Discovered in Release: 1.0.0

Overview

The purpose of the UFM cable validation tool is to validate the proper wiring of the network cluster and ensure high-quality links between the components.

The collector, also known as the bring up server, is the main component and is implemented as a docker container. It can be deployed on any machine connected to the management network of the switches, thereby facilitating communication with them. To manage large systems efficiently, an agent is installed on each switch, which is accountable for verifying the accuracy and quality of the switch links.

Collector

The collector is the main module that should be deployed and run on a host with management network access. It is important to note that an IB interface is not required on the host.

Deploying the Module

The Cable Validation tool can be deployed in two methods: as a standalone or as a UFM Enterprise plugin.

Deploying the Module as Standalone

Deploy the `cables_bringup` container on a host, as follows:

1. `docker load -i /tmp/cables_bringup_<version>.tar.gz`
2. `docker run --name cables_bringup -itd --network=host cables_bringup`
3. `docker exec -it cables_bringup /bin/bash`

Setting Docker Environment

Specifying the Network Interface

If the host system is equipped with multiple network interfaces and the switches are connected to the host through an interface that differs from the default management interface, the user can designate this particular interface by utilizing a specific environment variable, namely `AGENTS_IFC_NAME`. To illustrate, assuming the hypothetical interface name is `eno3`:

```
docker run --name cables_bringup -itd --network=host --env AGENTS_IFC_NAME=eno3 cables_bringup
```

Adding Hostnames

If the switches are not configured in the DNS server, you may add hostnames; the user may use the `--add-host` option when running the container. For example (assuming the switch name is `switch-3245fa` and its IP is 192.168.1.1):

```
docker run --name cables_bringup -itd --network=host --add-host=switch-3245fa:192.168.1.1 cables_bringup
```

Using Volumes

Volumes can be used for data persistence or easier file transfer to the `cables_bringup` container. The volume must be mapped to `/cable_bringup_root` in the container for data persistence. This volume can also be used for loading topology files. Example:

```
docker run --name cables_bringup -itd --network=host -v /opt/bringup_data:/cable_bringup_root cables_bringup
```

Overriding Apache Configuration

In the event that a host machine is running another Apache instance and utilizing the default ports 80/443, an alternative port may be designated for the bringup server by the user, these ports should be available and free. To accomplish this, the `APACHE_HTTPS_PORT` and `APACHE_HTTP_PORT` environment variables can be employed. Consider the following example:

```
docker run --name cables_bringup -itd --network=host --env APACHE_HTTP_PORT=9080 --env APACHE_HTTPS_PORT=9443
 cables_bringup
```

Deploying the Module as a UFM Enterprise Plugin

> ⚠ Please note that Running Cable Validation as plugin is not supported on UFM Gen2.0.

Deploy the module as a UFM Enterprise plugin as follows:

1. `docker load -i /tmp/cables_bringup_<version>.tar.gz`
2. `./manage_ufm_plugins.sh add -p cablevalidation`
3. `docker exec -it ufm-plugin-cablevalidation bash`

Copy Files to the Plugin

Users have two methods for copying files to the Cable Validation plugin:

1. Copy the files to the plugin's data volume located at `/opt/ufm/ufm_plugins_data/cablevalidation`, which is mapped to `/data/` inside the plugin container.
2. Use the '`docker cp`' command to transfer the required files directly to the container.

Overriding the Apache Configuration

When using Cable Validation as a plugin, the default ports 80/443 are already in use by UFM Enterprise. Therefore, port 8280 will be used for HTTP, and 8633 for HTTPS by default. Users can opt to use different ports for the bring-up server, provided that these ports are available and free.

The plugin `config.cfg` file can be modified to update `APACHE_HTTPS_PORT` and `APACHE_HTTP_PORT` variables for that purpose. To make this adjustment, follow these steps:

1. Execute `./manage_ufm_plugins.sh add -p cablevalidation` to add the Cable Validation plugin.
2. Stop the plugin using `./manage_ufm_plugins.sh stop -p cablevalidation`
3. Use `vim /opt/ufm/files/conf/plugins/cablevalidation/config.cfg` to modify the '`APACHE_HTTPS_PORT`' and '`APACHE_HTTP_PORT`' variables.
4. Update and save the file.
5. Start the plugin again with `./manage_ufm_plugins.sh start -p cablevalidation.`

With these changes, the new configuration will take effect, and Apache will run with the updated ports."

Running bringup CLI

1. Run `exec bringupcli` in the container:

```
docker exec -it cables_bringup bringupcli
```

2. Alternatively, it is possible to run exec bash in the container and run bringcli from anywhere within the container:

```
 docker exec -it cables_bringup
```

bringupcli Usage

`bringupcli` may have command line arguments, see usage below for more details:

```
root@r-ufm65:/# bringupcli -h

usage: bringupcli [-h] [-V] [-k]
```

Optional Arguments:

Argument	Description
`-h, --help`	Show this help message and exit
`-V, --version`	Show program version number and exit
`-k, --kill-other-sessions`	Kill other CLI sessions if existent

To initialize the tool, perform the following:

1. Load the fabric topology file:

```
load_topo <topo filename> topo file extension [cluster=<cluster name>]
load_ptp <topo filename > excel file extension [cluster=<cluster name>]
load_ip <ip filename> [cluster=<cluster name>]
load <topo filename> <ip filename>(both topo and ips) [cluster=<cluster name>]
```

```
load_clusters <clusters file>
```

2. Set the credentials for the switches. Use `set_default_creds/set_switch_creds` to set the credentials.
   The argument `` `[save=true|false] default: true` `` can be used with both commands to indicate whether to save the credentials to a file or not.
3. Deploy the agent on all switches. Run:

```
deploy_all_agents
```

Running bringup GUI

1. Open the following URL in the browser: https://<bringup_machine_ip>/cables_validation
2. Enter default credentials in the login page.
3. User management is not supported in the current version. To change it manually, use the `htpasswd` Linux utility.
   a. In the bringup container, locate the `.htaccees` file
   b. It is located at `${BRINGUP_CONF_APACHE_PATH}/.htaccess`
   c. Use `htpasswd` to add, modify or delete users.
4. user may change the default self signed certificate located by default in the container at:

```
SSLCertificateFile ${BRINGUP_CONF_APACHE_PATH}/certs/cv-cert.crt
SSLCertificateKeyFile ${BRINGUP_CONF_APACHE_PATH}/private/cv-cert.key
```

Update Certificate

To update a certificate, run the following command:

```
add_certificate <crt file> <key_file>: update the ssl certificate.
```

Validations

- `show_clusters`: Show list of loaded clusters as loaded from the clusters file.
- `show_switches`: Show list of loaded switches as loaded from the topology file
- `check_switch_status [cluster=<cluster>]`: Check switch connectivity status (Ping/JSON-API/Agent)
- `start_validation [cluster=<cluster>]`: Push topology to switches and get validation reports
- `stop_validation`: Unsubscribe from getting switches updates

Other Commands

- `show_switch_history`: Lists data files collected from switches in the last days
- `amber_show_latest`: Shows latest collected amber data from switches

Troubleshooting

- `deploy_single_agent`
- `deploy_all_agents`
- `remove_all_agents`
- `remove_single_agent`

Complete CLI Command Reference

1. `load_topo -` Loads topology file (topo file extension).
   `load_topo <filename> dns=true [cluster=<cluster name >]`-> assumes that DNS is active and you can access the switches by hostnames by default `dns=true`.
   A topo file example:

```
MQM8700 sw-hdr-proton01 CFG: main=4x
```

```
    P1 -4x-50G-> sw-hdr-proton02 P1
    P2 -4x-50G-> sw-hdr-proton02 P2
    P3 -4x-50G-> HCA_12 swx-proton03 mlx5_0/P1
    P4 -4x-50G-> HCA_12 swx-proton04 mlx5_2/P1
```

2. `load_ptp -` Loads PTP topology file (Excel file).
   `load_ptp <filename> sheets="sheet 1,my-sheet" dns=true [cluster=<cluster name >]`-> assumes that DNS is active and that you can access the switches by hostnames by the default setting of `dns=true`.
   If sheets argument is provided, only given sheets are loaded, otherwise, all sheets will be loaded. An example of sheet in the ptp file:

rack	U	Name	HCA/Port	Rack	U	Name	Port
316	22	c-csi-0329s	1	R113	22	c-csi-mqm9700-0327	1
316	24	c-csi-0331s	1	R113	22	c-csi-mqm9700-0327	1

   a. Please be aware that the designated port can be indicated either as a singular numerical value or as a combination of two numbers separated by a forward slash in the case of a split port. Concerning the port numbers for Host Channel Adapters (HCA), the following mapping convention should be applied: 1 represents `mlx5_0 P1`, 2 represents `mlx5_1 P1`, and so on.
   b. Moreover, it is mandatory for the Precision Time Protocol (PTP) file to incorporate a "Legend" sheet, which contains vital details regarding switch and host patterns. The below is an example:

Name	Model	Switch/HCA	Speed
c-csi-mqm*	MQM9700	switch	4x-100G
c-csi-0*	HCA_2	hca	4x-100G

3. `Load_ip [cluster=<cluster name >]`-Loads switch ip addresses, can be used if DNS is inactive. Loads the IP/switch-name mapping, to allow reaching the switch via REST API to retrieve local topology, GUID, etc. The file format is pairs of IP addresses and hostname. This file will be used in association with a ‘topo’ file in case DNS is unavailable.
   An IP file example:

```
# A comment
```

```
10.0.30  switch1
10.0.0.31  switch2
```

4. `load [cluster=<cluster name >]-` Loads both IP addresses and topo files. `load inputs/my-topo` loads `inputs/my-topo.topo` and `inputs/my-topo.ip`.
5. `load_clusters <clusters file> -` Clusters file should have the following format, where topo file should be in `xlsx` format and the ip file is optional, if it is not provided, `dns` will be considered as `true` when loading topo.

```
# cluster_name, topo_file, [ip_file]
CLUSTER1, cluster1_topo.xlsx, cluster1.ip
CLUSTER2, cluster2_topo.xlsx,
```

6. `show_switches [cluster=<cluster name >]-` Shows the list of loaded switches as loaded from the topology file. If the cluster name is provided, show the switch in the given cluster only.
Example output:

```
MQM8700 sw-hdr-proton01
-----------------------

MQM8700 sw-hdr-proton01 P3  --> swx-proton03 mlx5_0     P1
MQM8700 sw-hdr-proton01 P4  --> swx-proton04 mlx5_2     P1

MQM8700 ufm-sw-hdr01
--------------------

MQM8700 ufm-sw-hdr01 P1  --> ufm-sw-hdr02   P1
MQM8700 ufm-sw-hdr02
--------------------

MQM8700 ufm-sw-hdr02 P1  --> ufm-sw-hdr01   P1
```

7. `set_default_creds -` Sets the default switch/host credentials to override the built-in default credentials. These credentials are used for communication with any switch that does not have specific credentials.

```
set_default_creds user=<user> pwd=<pwd> [type=switch|host] [save=true|false]
```

8. `set_node_creds -` Sets the credentials for a specific switch/host, it can be used when the switch credentials are different than the defaults.

```
set_node_creds <switch> user=<user> pwd=<pwd> [save=true|false]
```

9. `deploy_all_agents` - Deploys agents on loaded switches that have no agents.
10. `deploy_single_agent` - Deploys agent on a specific switch.
11. `remove_all_agents` - Removes agents from loaded switches that have agents.
12. `remove_single_agent` - Removes an agent from a specific switch.
13. `show_switch_history` - Lists data files collected from switches in the last days `show_switch_history past=3d.` Past argument can be used to specify the history interval, by default it is set to one week `past=1w`.
14. `amber_show_latest` - Shows the latest collected amber data from switches
15. `check_switch_status [cluster=<cluster name>]` - Checks switch connectivity status (Ping/JSON-API/Agent). If the cluster is provided, the check will be done for the switches in the provided cluster only.
    Example output:

```
Host IP                ping    JSONAPI    Agent
-----------------------------   -------------    ----       ----     -----
sw-hdr-proton01.mtr.labs.mlnx   209.44.74        True       True     True
ufm-sw-hdr01.mtr.labs.mlnx      10.209.36.113    True       True     True
ufm-sw-hdr02.mtr.labs.mlnx      10.209.36.122    True       True     True
```

16. `add_certificate <crt file> <key_file>` - Updates the SSL certificate file used by Apache for secure connections. The provided file should be a valid SSL certificate file in crt format. The old certificate file will be backed up before replacing it with the new one.
17. `start_validation` - Initiates validation routine: pushes topology to switches and gets validation reports timeout (an optional argument), in which validation stops. (For example `timeout=20m` or `timeout=2h`). If timeout is not provided, use the `stop_validation` command to stop it. `start_validation timeout=n` (in seconds/minutes/hours/days).
18. `stop_validation` - Stops validation routine. Unsubscribe from getting switches updates.
19. `version` - Shows application version.
20. `exit` - Exits the application.
21. `help` - Shows a list of commands. For help on a specific command, run `help <command>`

Bringup Server REST API

The collector has a web server listening on two internal ports 8251 and 8252. These ports are not advertised outside the machine. The bringup server is running on the *Apache* server which uses the default http/https ports. It is not recommended to change the internal ports, as this requires changing the

Apache service configuration. The *Apache* service uses a self signed certificate, that the user can change to his own certificate. All REST APIs can run only with https.

> ⚠ Please note that for all the following REST API URLs, the <host> attribute is the host IP or the hostname with the correct port number in case it is not the default one. For example:
> - https://10.20.30.40:8633/
> - https://10.20.30.40/ # the default port: 433
> - https://server-name:8639/

The following are the supported REST APIs:

Login

To use a REST API, you need to have session credentials. If you want to use curl to access the REST API, you should log in first by going to the URL cablevalidation/login and saving the cookie. After that, you can use the saved cookie for subsequent requests.

```
# login and save cookie
curl -k -X POST -c cookies.txt -d "httpd_username=<user>" -d "httpd_password=<password>" https://<host>/
cablevalidation/login
# use saved cookie for REST API requests
curl -k --cookie cookies.txt https://127.0.0.1/cablevalidation/report/validation
```

Retrieving Validation Report

Run:

```
GET https://<host>/cablevalidation/report/validation
```

Validation Report Output Example

```
curl -k https://swx-proton01/cablevalidation/report/validation | python3 -m json.tool
{
    "report": "ValidationReport",
    "stats": {
        "in_progress": 3,
        "no_issues": 0,
        "not_started": 0
    },
    "issues": [
        {
            "timestamp": 1666176949.5110743,
            "node_desc": "MQM8700 sw-hdr-proton01",
            "issues": [
                [
                    "Wrong-neighbor",
                    "MQM8700 sw-hdr-proton01:P3",
                    "HCA_12 swx-proton03 mlx5_0:P1",
                    "None:PNA"
                ],
                [
                    "Wrong-neighbor",
                    "MQM8700 sw-hdr-proton01:P4",
                    "HCA_12 swx-proton04 mlx5_2:P1",
                    "HCA_12 swx-proton04 mlx5_0:P1"
                ]
            ]
        },
        {
            "timestamp": 1666176949.4999607,
            "node_desc": "MQM8700 ufm-sw-hdr02",
            "issues": [
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr02:P2",
                    "NONE",
                    "MQM8700 ufm-sw-hdr01:P2"
                ],
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr02:P3",
```

```
                    "NONE",
                    "MQM8700 ufm-sw-hdr01:P3"
                ],
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr02:P7",
                    "NONE",
                    "MQM8700 ufm-sw-hdr01:P7"
                ]
            ]
        },
        {
            "timestamp": 1666176949.4870453,
            "node_desc": "MQM8700 ufm-sw-hdr01",
            "issues": [
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr01:P2",
                    "NONE",
                    "MQM8700 ufm-sw-hdr02:P2"
                ],
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr01:P3",
                    "NONE",
                    "MQM8700 ufm-sw-hdr02:P3"
                ],
                [
                    "Extra-cable",
                    "MQM8700 ufm-sw-hdr01:P7",
                    "NONE",
                    "MQM8700 ufm-sw-hdr02:P7"
                ]
            ]
        }
    ]
}
```

Bringup Commands Support via REST API

The processing of bringup commands is not limited to the CLI; it can also be accomplished through the REST API.

Processing a Command

Run:

```
POST https://<host>/cablevalidation/commands/{command_name} <command-data>
```

Supported Commands

Command	Async	Argument	Type	Mandatory
load_topo	False			
		dns	bool	False
		files	list	True
		cluster	str	False
load_ip	False			
		files	list	True
		cluster	str	False
load_ptp	False			

Command	Async	Argument	Type	Mandatory
		dns	bool	False
		sheets	list	False
		files	str	True
		cluster	str	False
load	False			
		file_prefix	str	True
		cluster	str	False
Load_clusters	False			
		file	str	True
set_default_creds	False			
		user	str	True
		pwd	str	True
		type	str	False
		save	bool	False
set_node_creds	False			
		user	str	True

Command	Async	Argument	Type	Mandatory
		pwd	str	True
		type	str	True
		save	bool	False
deploy_all_agents	True			
deploy_single_agent	True			
		switch	str	True
remove_all_agents	True			
remove_single_agent	True			
		switch	str	True
start_validation	True			
		cluster	str	False
stop_validation	True			
add_certificate	False			
		crt_file	str	True
		key_file	str	True
check_switch_status	True			
show_switches	False			

Command	Async	Argument	Type	Mandatory
		name_pattern	str	False
show_switch_history	False			
		switches	str	False
		past	str	False
amber_show_latest	False			
		filter	str	False
exit	False			

Process Command Example

The command body is a JSON dictionary of key-value arguments as described in the table below.

```
curl -k https://127.0.0.1/cablevalidation/commands/load_topo -d '{"files":["inputs/lab.topo"], "dns":true}' -X
POST
Command load_topo completed successfully
```

Supported Commands

Getting Command Output

```
GET https://<host>/cablevalidation/commands/{command_name}/output
```

`timestamp` is an optional argument that enables the user to obtain only the output generated after a particular point in time. It is included in the following format: `GET https://<host>/cablevalidation/commands/{command_name}/output?timestamp=<val>`.

The response to this request takes the form of a JSON dictionary, containing the following details:

1. `command`: the processed command.
2. `request_ts`: timestamp of the request made by the user, if a timestamp was provided; otherwise, it is set to 0.
3. `last_ts`: the timestamp of the most recent message in the output, which the user can utilize for subsequent requests.
4. `status`: represents the current status of the command, which can be either "Completed" or "InProgress".
5. `content`: the actual output log of the command.

Command Output Example

```
curl -k https://localhost/cablevalidation/commands/deploy_all_agents/output 2> /dev/null | python -m json.tool
{
    "command": "deploy_all_agents",
    "content": [
        "Will install agent on 10.209.44.74",
        "Will install agent on 10.209.36.113",
        "Will install agent on 10.209.36.122",
```

Getting Commands Processing Status

```
GET https://<host>/cablevalidation/commands/status
```

The response to the request provides a JSON dictionary that conveys pertinent information regarding the processing status of commands, which may fall into one of two categories:

- Idle - In this scenario, the user is at liberty to initiate a new command.
- Executing <command> - In this instance, the processor is currently engaged in executing a command, and as such, is incapable of processing any new commands until the current operation is complete.

Getting a List of Supported Commands

The following command returns a JSON dictionary with all supported commands as well as their arguments and if it async or sync.

```
GET https://<host>/cablevalidation/commands
```

Supported Commands Output Example

Output has been cut.

```
{
    "load_topo": {
        "args": {
            "dns": {
                "type": "bool",
                "mandatory": false
            },
            "files": {
                "type": "list",
                "mandatory": true
            }
```

```
        },
        "is_async": false
    }
}
```

Getting Help on Command

```
GET https://<host>/cablevalidation/commands/{command_name}/help
```

The response to the request is in the form of a JSON dictionary, which provides the following details:

- `command`: The name of the command that was executed.
- `help`: A list of output lines that convey relevant information about the command.

Command Help Example

```
curl -k https://localhost/cablevalidation/commands/load_topo/help 2> /dev/null | python -m json.tool
{
    "command": "load_topo",
    "help": [
        "",
        "        load_topo filename dns=true/false",
        "",
        "        default dns=true",
        "        If no dns server to resolve hostnames in topo file, you should set dns=false and provide IP
addresses file.",
        "        when true, no need to provide IP addresses.",
        ""
```

Rack View

Rack and unit information can be shown when loading a PTP Excel file, however, topo files do not contain such information, therefore, rack view is not available.

Rack view is supported via two REST APIs.

Getting List of Racks

The following command returns a JSON list of all loaded racks.

```
GET https://<host>/resources/racks
```

Racks List Output Example

```
[
    "1108",
    "1106"
]
```

Getting Rack View of a Specific Rack

The following command returns a JSON dictionary with rack details.

```
GET https://<host>/resources/racks/{rack-name}
```

Rack View Output Example

```
{
    "name": "1108",
    "units": [
        {
            "nodedesc": "MSB7800 r-ufm-sw10",
            "ports": [
```

```
                {
                    "port": "P25",
                    "syndrome": "Wrong-neighbor"
                },
                {
                    "port": "P26",
                    "syndrome": "Wrong-neighbor"
                },
                {
                    "port": "P27",
                    "syndrome": "Active"
                },
                {
                    "port": "P28",
                    "syndrome": "Active"
                }
            ],
            "unit": "40"
        }
    ]
}
```

Cables Agent

The Cables agent is implemented as a docker container that executes on the switch to gather data on neighboring switches and link quality. The agent operates a web service capable of providing information on ports and links through user queries. Moreover, the agent transmits validation reports to the bringup server.

Check if Cable Agent is Running

Check if cable agent is running on the switch:

1. Run:

```
ssh admin@<switch-ip-or-name>
```

2. Enable
3. Show docker images
4. Exit

If cables agent is running on the switch, the following output is prompted.

```
---------------------------------------------------------------------------
Image                                Version      Created            Size
---------------------------------------------------------------------------
cables_agent                         latest       13 hours ago       788MB
```

Deploying Cable Agent on the Switch

Usually, it is not necessary to manually deploy the agent onto the switch, as it is recommended to use the `deploy_all_agents` or `deploy_single_agent` commands from the bringup CLI. However, in instances where manual deployment is required, the following commands can be executed:

1. `enable`
2. `configure terminal`

3. no docker shutdown
4. image fetch scp://<user>:<pwd>@<hostname>/tmp/cables_agent_<version>.tar.gz cables_agent_latest.tar.gz
5. docker load cables_agent_latest.tar.gz
6. docker start cables_agent latest cables_agent now-and-init privileged network

For cleanup, run:

1. docker no start cables_agent
2. docker remove image cables_agent latest
3. image delete cables_agent_latest.tar.gz

To enter terminal in the container running on the switch, run:

1. enable
2. configure terminal
3. docker exec cables_agent /bin/bash

Cables Agent REST API

The agent has a web server listening on port 8251. The following two REST APIs are supported:

1. https://<switch-ip-or-name>:8251/resources/links
2. https://<switch-ip-or-name>:8251/resources/ports

Links Output Example

```
curl -k https://sw-hdr-proton01:8251/resources/links | python3 -m json.tool
[
 {
    "info": {
        "md5": "256477d766fa8d8853848c43c35982ba",
        "timestamp": 1659355401394591,
        "time": "2022-08-01 12:03:21.394601"
```

```
    },
    "src": {
        "Node Description": "MF0;sw-hdr-proton01:MQM8700/U1",
        "Guid": "0x0c42a1030079a6ec",
        "ip": "10.209.44.74",
        "Node Name": "sw-hdr-proton01"
    },
    "dests": {
        "4": {
            "Node Description": "swx-proton04 mlx5_2",
            "Guid": "0xb8cef6030083bea2",
            "LocalPort": "1"
        },
        "2": {
            "Node Description": "Quantum Mellanox Technologies",
            "Guid": "0xb8cef60300fbf210",
            "LocalPort": "2"
        },
        "3": {
            "Node Description": "swx-proton03 mlx5_0",
            "Guid": "0xb8cef6030083bf02",
            "LocalPort": "1"
        },
        "1": {
            "Node Description": "Quantum Mellanox Technologies",
            "Guid": "0xb8cef60300fbf210",
            "LocalPort": "1"
        }
    }
}
]
```

Output Example of Ports

```
curl -k https://sw-hdr-proton01:8251/resources/ports | python3 -m json.tool
[
    {
```

```
        "port": "IB1/10",
        "port_num": "10",
        "logical": "Down",
        "physical": "Polling"
    },
    {
        "port": "IB1/11",
        "port_num": "11",
        "logical": "Down",
        "physical": "Polling"
    },
    {
        "port": "IB1/12",
        "port_num": "12",
        "logical": "Down",
        "physical": "Polling"
    },
    {
        "port": "IB1/13",
        "port_num": "13",
        "logical": "Down",
        "physical": "Polling"
    }
]
```

Document Revision History

Revision	Date	Description
Rev 1.2	Nov 5, 2023	Updated the following sections: • Deploying the Module - Added instructions on Deploying the Module as Standalone and Deploying the Module as a UFM Enterprise Plugin • bringupcli Usage - Updated commands • Complete CLI Command Reference - Added `add_certificate <crt file> <key_file>` command • Bringup Server REST API - Added a note on REST API URLs and changed all IP examples to `<host>` instead of `<host-ip-or-name>` • Supported Commands - Added the `add_certificate` command Added the following section: • Update Certificate
Rev 1.1	Aug 10, 2023	Updated the following sections: • bringupcli Usage - Added cluster option in load commands and documentation for save arguments • Validations - Added `show_clusters` and cluster arguments • Complete CLI Command Reference - Added cluster arguments and "load_clusters <clusters file>" CLI command • Supported Commands - Updated table with new cluster arguments
Rev 1.0	May 8, 2023	First release

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

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

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

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

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

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

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT,

INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.