NVIDIA DOCA Management Service Guide
Table of contents

Introduction
Requirements
Service Deployment
Configuration
  General Flags
  Security Flags
  Provisioning Flags
Description
  gNMI Command
  Get Supported Paths
  Get Request
  Set Request
  gNOI Commands
    OS
    System
This guide provides instructions on how to use the DOCA Management Service on top of NVIDIA® BlueField® Networking Platform or ConnectX® Network Adapters.

**Note**

DOCA DMS service is currently supported at Alpha level.

**Introduction**

DOCA Management Service (DMS) is a one-stop shop for the user to configure and operate NVIDIA BlueField and ConnectX devices. DMS governs all scripts/tools of NVIDIA with an easy and industry-standard API created by the OpenConfig community. The user can configure BlueField or ConnectX for any mode whether locally (ssh) or remotely (grpc). It makes it easy to migrate and bootstrap any customer for any NVIDIA network device.

DMS exposes configurable BlueField/ConnectX parameters over the external interface to support a management station in an automated configuration of the NVIDIA Network Adapters. The exposed interface presents a uniform approach for BF/CX device configuration and keeps hidden details about the internal tools used for the configuration of BlueField or ConnectX features.

The DMS is a Client-Server architecture. Using a daemon, the service handles the discovery of resources, and is ready to receive commands from clients, the user can use DMSc (DMS Client) which delivers as part of the DMS, or use/create any other client.

**Info**

Please refer to the OpenConfig site for an explanation of the OpenConfig protocol.

The Yang models describe a config tree which is easy to navigate and find any "config leaf" using XPath capabilities. Most gNMI/gNOI protocols are common with the
OpenConfig community, utilizing gRPC protocol for transferring the command.

### Note

The DOCA Yang model is experimental.

### Note

The gNMI Subscribe mechanism for streaming telemetry is not currently supported yet.

### Info

DMS can run either on the host machine where BlueField or ConnectX devices are installed or on BlueField Arm itself (when BlueField is operating in DPU mode).

### Requirements

DMS requires DOCA to be installed on the target system, where DMS Service will be running:

- DMS for Host - requires DOCA for Host package to be installed on the host system (with doca-networking or doca-all profiles).
- DMS for DPU (BlueField Arm) - requires DOCA Image to be installed on BlueField Arm.

Please follow these instructions to install DOCA: [NVIDIA DOCA Installation Guide for Linux](#).
Service Deployment

DMS has 3 major components:

- **DMSD – Server** – DMS server inside the BlueField or on the host with an NVIDIA PCIe device

- **DMSC – Client** – DOCA provides OpenConfig client. Customers can choose to use this client, any other open-source client, or develop their own (gRPC-based) client.

- **Yang files** – Yang model files contain the data model used to configure the BlueField device, NVIDIA-specific extension to common OpenConfig YANG Models.

OpenConfig consists of 2 main protocols:

- **gNMI** – gRPC Network Management Interface, protocol to configure of network device.

- **gNOI** – gRPC Network Operations Interface, a protocol to perform operational commands on network device (i.e., provision, upgrade, reboot).

The following is an architectural diagram of DMS:
The following diagram presents the DMS mode of operation, as the DMS client can operate from anywhere:

1. Both DMS client and server components are deployed on the Host
2. Both DMS client and server components are deployed on DPU (BlueField Arm)
3. DMS server component is deployed on the Host, while DMS client is deployed remotely (connecting to DMS server over management network)
4. DMS server component is deployed on DPU (BlueField Arm), while DMS client is deployed remotely (connecting to DMS server over management network)

DMSD is a systemd service installed on the DPU by default with the BFB-Bundle and can be enable/disabled using systemctl. DMSD can be accessed using the command dmscli and
provided the `dmsd` user password (default is the root OS password). A systemd template is provided on host packages.

**Configuration**

To see the full list of flags, use the help flag (i.e., `dmsd -help`, `dmsd -h`).

**General Flags**

- `-bind_address <string>` – Bind to `<address>:<port>` or just `<port>` (default is :9339). Can be localhost for local use case, or an IP address for remote use case.
- `-v <value>` – log level for V logs
- `-target_pci <string>` – The target PCIe address (i.e., 03:00). Auto-select if only one NVIDIA network device is present; otherwise, the PCIe address must be specified.

**Security Flags**

- `-auth string` – this flag has 3 options:
  - **Shadow**
    - Zero-touch, admin not required to create any dedicated additional user for DMS (re-use OS user)
    - Read the hashed password in real time on each client request
    - Use flags `-username -shadow`
    - Example: `-username root -shadow /etc/shadow/`
    - To disable: `-noauth` flag
  - **Credentials**
    - Admin must set a strong password
    - Use flags `-username -password`
- Example: -username root -password 123456
- To disable: -noauth flag
- Can leave password flag empty to invoke prompt for password at demon boot

**Certificate File**
- The most secure option, based on (m)TLS
- Example: -ca /tmp/ca.crt -ca_key /tmp/ca.key
- To disable: -notls option

**Provisioning Flags**
- `-target_pci <string>` – The target PCIe address (i.e., 03:00). Auto-select if only one NVIDIA network device is present; otherwise, the PCIe address must be specified.
- `-image_folder <string>` – Specify image install folder. Can copy images directly to the folder to avoid transfer over the net. Default create folder: `/tmp/dms`.
- `-chunk_size_ack <uint>` – The chunk size of the image to respond with a transfer response in bytes (default: 12000000)
- `-exec_timeout <uint>` – The maximum execution timeout in seconds for a command if not responding (not printing to stdout); 0 (default) is unlimited

**Description**

**gNMI Command**

In DMSC, the gNMI part is powered by the GNMIC project.
Prompt mode with autocomplete options can be invoked using the command `prompt`. It can be accessed using the command `dmscli` and provided the `dmsd` user password (default is the root OS password).

### Get Supported Paths

```
dmsc -a localhost:9339 -u root -p <password> --file /opt/mellanox/doca/service/dms/yang <command>
```

For more information, please refer to GNMIC documentation.

```
dmsc --file /opt/mellanox/doca/service/dms/yang path --types --descr
```

`/interfaces/interface[name=\*]/config/enabled` (type=boolean)
This leaf contains the configured, desired state of the interface.

Systems that implement the IF-MIB use the value of this leaf in the 'running' datastore to set IF-MIB.ifAdminStatus to 'up' or 'down' after an ifEntry has been initialized, as described in RFC 2863.

Changes in this leaf in the 'running' datastore are reflected in ifAdminStatus, but if ifAdminStatus is changed over SNMP, this leaf is not affected.

`/interfaces/interface[name=\*]/config/mtu` (type=uint16)
Set the max transmission unit size in octets for the physical interface. If this is not set, the mtu is set to the operational default -- e.g., 1514 bytes on an Ethernet interface.

`/interfaces/interface[name=\*]/config/type` (type=identityref)
The type of the interface.

When an interface entry is created, a server MAY initialize the type leaf with a valid value, e.g., if it is possible to derive the type from the name of the interface.
If a client tries to set the type of an interface to a value that can never be used by the system, e.g., if the type is not supported or if the type does not match the name of the interface, the server MUST reject the request.

A NETCONF server MUST reply with an rpc-error with the error-tag 'invalid-value' in this case.

/interfaces/interface[name=*]/ethernet/nvidia/config/inter-packet-gap  (type=uint8)
   Inter packet gap configuration, in 4B unit
/interfaces/interface[name=*]/ethernet/nvidia/config/rate-limit (type=uint16)
   The percentage of bandwidth, in permile units, to be used on the port.
/interfaces/interface[name=*]/name  (type=leafref)
   References the name of the interface
/interfaces/interface[name=*]/nvidia/cc/config/priority[id=*]/id  (type=leafref)
/interfaces/interface[name=*]/nvidia/cc/config/priority[id=*]/np_enabled  (type=boolean)
   Enable CC NP for a given priority on the interface
/interfaces/interface[name=*]/nvidia/cc/config/priority[id=*]/rp_enabled  (type=boolean)
   Enable CC RP for a given priority on the interface
/interfaces/interface[name=*]/nvidia/cc/slot[id=*]/config/enabled  (type=boolean)
   Enable a CC algo slot execution.
/interfaces/interface[name=*]/nvidia/cc/slot[id=*]/id  (type=leafref)
   CC algo slot ID.
/interfaces/interface[name=*]/nvidia/cc/slot[id=*]/param[id=*]/config/value  (type=algo_param_value)
   Parameter value within the CC algo slot.
/interfaces/interface[name=*]/nvidia/cc/slot[id=*]/param[id=*]/id  (type=leafref)
   Parameter ID within the CC algo slot.
/interfaces/interface[name=*]/nvidia/qos/config/pfc  (type=boolean)
   Enables PFC
/interfaces/interface[name=*]/nvidia/qos/config/priority[id=*]/id  (type=prio)
   Priority id.
/interfaces/interface[name=*]/nvidia/qos/config/trust-mode  (type=identityref)
   Trust mode for the interface QoS.
/interfaces/interface[name=*]/nvidia/roce/config/adaptive-retransmission  (type=boolean)
   Enable adaptive retransmission
/interfaces/interface[name=*]/nvidia/roce/config/adaptive-routing-force (type=boolean)
   Force adaptive routing even if feature was not negotiated between a requestor and responder.
/interfaces/interface[name=*]/nvidia/roce/config/rtt-resp-dscp  (type=uint8)
   Defines the DSCP fixed value used if mode is set to FIXED.
/interfaces/interface[name=*]/nvidia/roce/config/rtt-resp-dscp-mode  (type=identityref)
   Defines the method for setting DSCP in RTT response packets.
/interfaces/interface[name=*]/nvidia/roce/config/slow-restart  (type=boolean)
   Enable slow restart when congestion
/interfaces/interface[name=*]/nvidia/roce/config/slow-restart-idle  (type=boolean)
   Enable slow restart when idle
Get requests happen in real-time without cache. Get command require providing the Yang Xpath as described in the following:

dmsc <flags> get --path /interfaces/interface[name=p0]/config/mtu
[
   {
      "source": "localhost:9339",
      "timestamp": 1712485149723248511,
      "time": "2024-04-07T10:19:09.723248511Z",
      "updates": [
         {
            "Path": "interfaces/interface[name=p0]/config/mtu",
            "values": {
               "interfaces/interface/config/mtu": "1500"
            }
         }
      ]
   }
]
Info

To insert params in the path, as an indication of the interface name (p0).

Set Request

Set requests happen immediately, invoking tools to configure the OS.

Set commands require providing Yang Xpath as described in the following:

```
dmsc <flags> set --update /interfaces/interface[name=p0]/config/mtu::int::9216
{
  "source": "localhost:9339",
  "time": "1970-01-01T00:00:00Z",
  "results": [
    {
      "operation": "UPDATE",
      "path": "interfaces/interface[name=p0]/config/mtu"
    }
  ]
}
```

Info

To insert params in the path, as an indication of the interface name (p0).

Note
It is also possible to invoke a command JSON list:

dmsc <flags> set --request-file req.json

req.json example:

```json
{
  "updates": [
    
    {
      "path": "/interfaces/interface[name=p0]/config/mtu",
      "value": 9216,
      "encoding": "uint"
    },
    
    {
      "path": "/interfaces/interface[name=p0]/config/mtu",
      "value": true,
      "encoding": "bool"
    }
  ]
}
```
gNOI Commands

In DMSc, the gNOI part is powered by GNOIC project, for full docs refer to GNOIC docs.

```
dmsc -a localhost --port 9339 --tls-cert client.crt --tls-key client.key <command>
```

Prompt mode with autocomplete options can be invoked using the command `prompt`.

All commands are blocking unless specified otherwise.

**OS**

The following subsections present actions for provisioning a new DOCA Image (BFB) or firmware on BlueField.

**Install**

This command transmits the file from the client to the server and authenticates the file's validity:

```
dmsc <flags> os install --version <free_text_version> --pkg <bfb|cfg|fw path>
dmsc <flags> os install --version 2.7.0 --pkg DOCA_2.7.0_Ubuntu.bfb
dmsc <flags> os install --version 2.7.0 --pkg config.cfg
```

The file is saved to the folder specified in the `-image_folder` flag (default `/tmp/dms`) if the file authenticates successfully. The file's extension is autodetected and is written automatically if `none` is provided in the `--version` field. Users may copy the file to the folder manually and invoke the command with file extension to authenticate the file. No file
transfer is initiated if the file already exists in the folder and the version specified with the extension.

### Activate

Activate the command deploy the BFB bundle/firmware to the hardware:

```
dmsc <flags> os activate --version 2_7_0 # Invoke all files under 2_7_0 name
dmsc <flags> os activate --version "2_7_0.bfb;0_0_1.cfg;24_29_0046.fw"
```

The `--version` flag provides a version to search for in the folder specified by the `-image_folder` flag (default `/tmp/dms`). If no extension is provided, the command uses all files under the version name.

To activate separate files, use the `--version` flag separated by semi-colon.

#### Note

After running the command to activate firmware, firmware reset is automatically invoked.

### Verify

Verify command retrieves the firmware and BFB bundle version:

```
dmsc <flags> os verify
```

The return value consists of both versions separated by semi-colon.
System

The following subsections provide actions for rebooting the BFB bundle/firmware on the BlueField.

Reboot Status

To verify BFB is rebooting:

```
dmsc <flags> system reboot-status
```

The value returned is false if the system is active. It is true if the system is rebooting. If the status cannot be retrieved, the status appears as a failure and the message field indicates what the issue is.

The flag --reboot_status_check <string> checks if firmware reboot is needed:

- If set to fast (default), a quick test occurs but not accurate (any config can trigger this flag)
- If set to strict, a more accurate test occurs but slower
- If set to none, then firmware check is skipped

Reboot

Note

Currently, the BFB bundle can only be retrieved if it was installed via DMS.
To reboot the BlueField Arm and firmware:

```
dmsc <flags> system reboot --delay <uint>s --subcomponent <string> --method <string>
```

This command is non-blocking and returns immediately.

The flag `--delay` specifies the time interval to wait before invoking the reset.

The subcomponent and method are optional. By default, the reboot executes with the lowest reset level and type available.

**Note**

Currently, DMS supports `--subcomponent ARM --method <WARM|POWERDOWN>` flags.