# NVIDIA DOCA URL Filter

Application

MLNX-15-060477 _v1.4 | September 2022

# Table of Contents

# Chapter 1. Introduction

URL filtering limits access by comparing web traffic against a database to prevent users from different threats, malware and accessing harmful sites such as phishing pages.

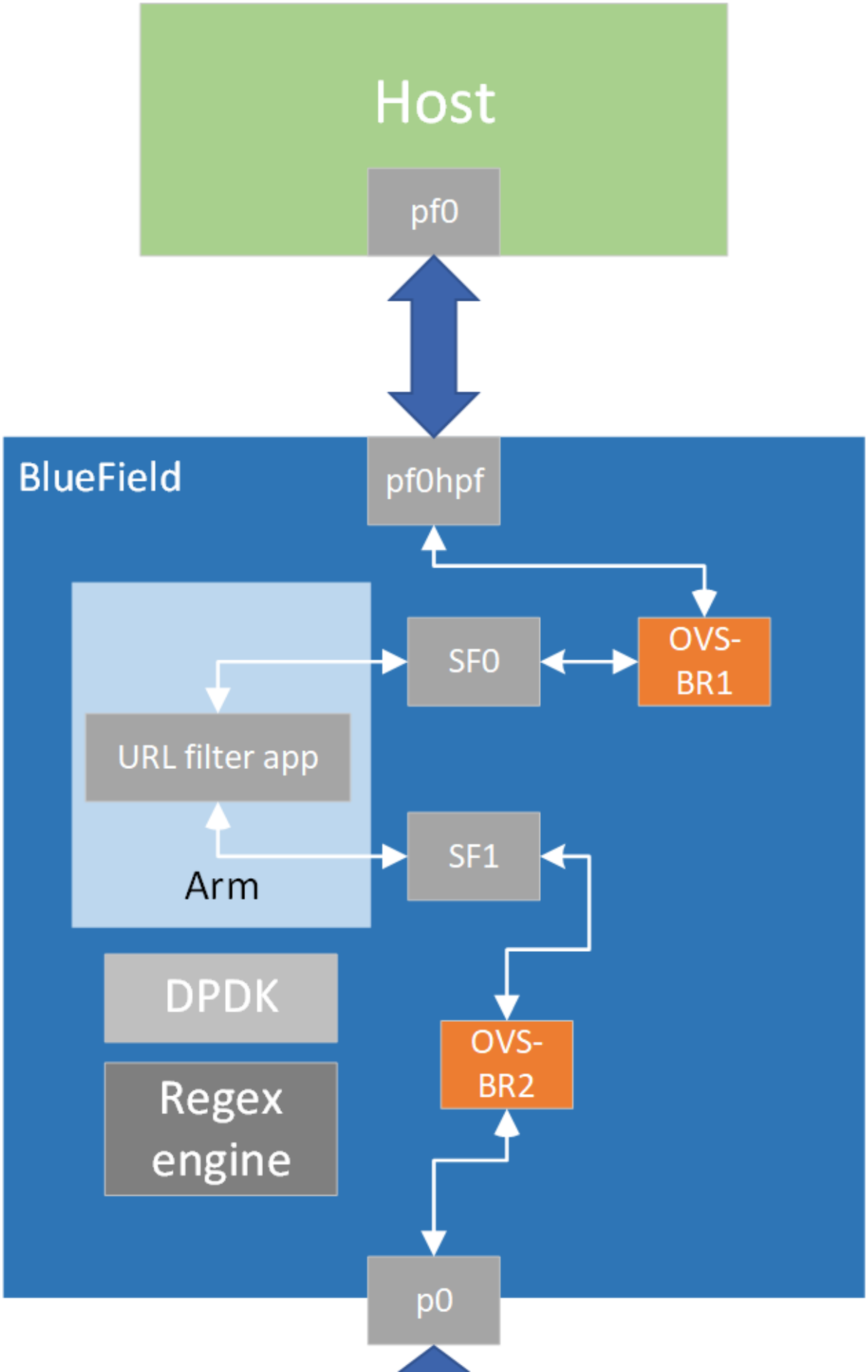
This kind of content filtering can increase network security and enforce policies on different network resources.

URL filter uses Suricata format to create a signature file that can be compiled and loaded to the DPI RegEx engine. The signatures created attempt to find a match in a TCP header's payload, either inside an HTTP header or an SNI match inside a TLS header. Two different signatures. In terms of packet direction, the match can be both on the sender and receiver. The exact format can be seen in the attached signature file.

# Chapter 2. System Design

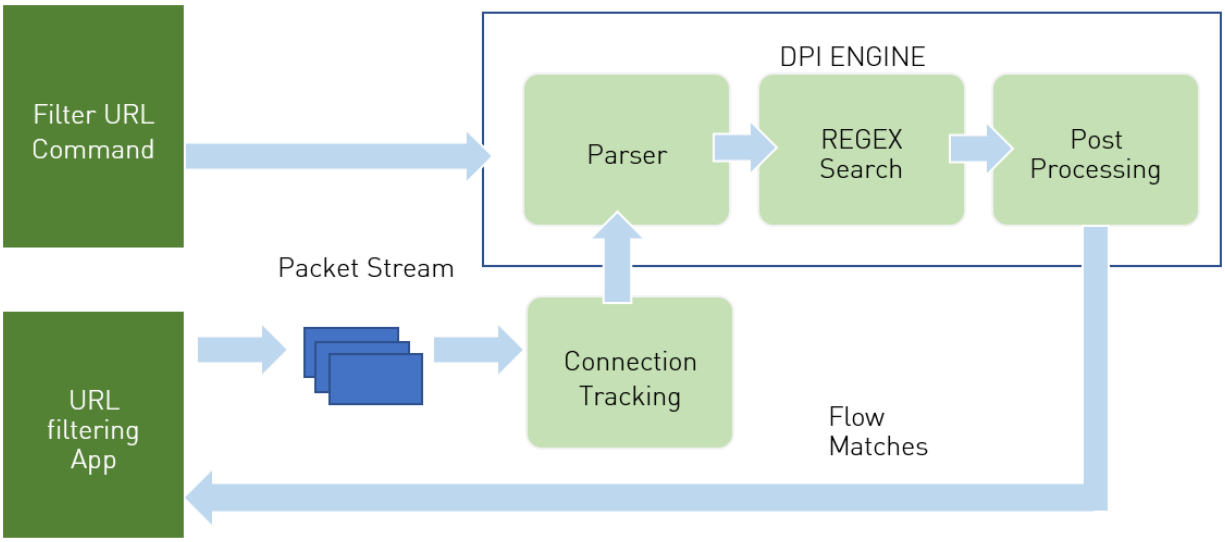
URL filtering is designed to run as "bump-on-the-wire" on the BlueField-2 instance. It intercepts traffic coming from the wire and passes it to the physical function (PF) representor connected to the host.

URL filter utilizes the SFT and RegEx engines which are HW accelerators on BlueField.


Host
pf0
BlueField
pf0hpf
SF0
OVS-
BR1
URL filter app
SF1
Arm
DPDK
OVS-
BR2
Regex
engine
p0

# Chapter 3. Application Architecture

- User adds a URL using the URL command from the CLI. Available commands:
  - Create database – create DB of URLs to be filtered (also removes old signatures)
  - Filter – add specific URL/URL RegEx pattern to be filtered
  - Commit database [PATH] – compile and load signature database
- Ingress traffic is identified using the connection tracking module
- Traffic is scanned against compiled signature DB
- Post-processing is performed for match decision
- Matched flows are filtered and traffic is blocked




1. Signatures are compiled by the DPI compiler and are then loaded to the DPI engine.
2. Ingress traffic is identified and classified using the stateful table module in the DPDK libs which utilizes the connection tracking hardware offloads. This allows flow classifications to be done at the hardware level and forwarded to a hairpin queue without being processed by the software, which increases performance dramatically.
3. Traffic is scanned against the DPI engine compiled signature DB.
4. Post-processing is performed for match decision.

5. Matched flows are identified and can be offloaded to the hardware to increase performance as not further inspection is needed.
6. Flow termination is done by the aging timer (set in the SFT to 60 seconds). When a flow is offloaded it cannot be tracked and destroyed.

**Note:** It is important to note that only sites that support non-encrypted HTTP traffic can be matched against signatures created by the URL filtering as it specifically targets the URI field in the HTTP request.

# Chapter 4. DOCA Libraries

This application leverages the DOCA DPI library.

# Chapter 5. Configuration Flow

1. Parse application argument.
   ```
   doca_argp_init();
   ```
   a). Initialize the arg parser resources.
   b). Register DOCA general flags.
   ```
   register_url_params();
   ```
   c). Register URL filter application flags.
   ```
   doca_argp_start();
   ```
   d). Parse DPDK flags and invoke handler for calling the `rte_eal_init()` function.
   e). Parse app flags.
2. DPDK initialization.
   ```
   dpdk_init();
   ```
   Calls `rte_eal_init()` to initialize EAL resources with the provided EAL flags.
3. DPDK port initialization and start.
   ```
   dpdk_queues_and_ports_init();
   ```
   a). Initialize SFT.
   b). Initialize DPDK ports, including mempool allocation.
4. URL filter initialization.
   ```
   url_filter_init();
   ```
   a). Init URL filter resources.
   b). Configure RegEx engine.
   c). Configure DPI queues.
5. Configure DPI packet processing.
   ```
   dpi_worker_lcores_run();
   ```
   a). Configure DPI enqueue packets.
   b). Send jobs to RegEx engine.
   c). Configure DPI dequeue packets.
6. Add URL shell command
   ```
   initiate_cmdline("URL FILTER>>");
   ```
   a). Create database.
   b). Filter http.
   c). Commit database.

7. DPI destroy
   ```
   url_filter_destroy();
   ```
   - Stop DPI worker and free DPI resources
8. DPDK ports and queues destruction.
   ```
   dpdk_queues_and_ports_fini();
   ```
9. DPDK finish.
   ```
   dpdk_fini();
   ```
   calls `rte_eal_destroy()` to destroy initialized EAL resources.
10. Arg Parser destroy.
    ```
    doca_argp_destroy();
    ```
    - Free DPDK resources

# Chapter 6. Running Application

1. Refer to the following documents:
   - NVIDIA DOCA Installation Guide for details on how to install BlueField-related software.
   - NVIDIA DOCA Troubleshooting Guide for any issue you may encounter with the installation, compilation, or execution of DOCA applications.
2. The URL filtering example binary is located under `/opt/mellanox/doca/applications/url_filter/bin/doca_url_filter`. To build all the applications together, run:
```
cd /opt/mellanox/doca/applications/
meson build
ninja -C build
```
3. To build the URL Filter application only:
   a). Edit the following flags in `/opt/mellanox/doca/applications/meson_option.txt`:
      - Set `enable_all_applications` to `false`
      - Set `enable_url_filter` to `true`
   b). Run the commands in step 2.

   **Note:** `doca_url_filter` is created under `./build/url_filter/src/`.

   Application usage:
```
Usage: doca_url_filter [DPDK Flags] -- [DOCA Flags] [Program Flags]
DOCA Flags:
  -h, --help                    Print a help synopsis
  -v, --version                 Print program version information
  -l, --log-level               Set the log level for the app <CRITICAL=0,
 DEBUG=4>
Program Flags:
  -p, --print-match             Prints FID when matched in DPI engine
  -f, --fragmented              Enables processing fragmented packets
```

   **Note:** For additional information on available flags for DPDK, use `-h` before the `--` separator:
```
/opt/mellanox/doca/applications/url_filter/bin/doca_url_filter -h
```

   **Note:** For additional information on the application, use `-h` after the `--` separator:
```
/opt/mellanox/doca/applications/url_filter/bin/doca_url_filter -- -h
```
4. Running the application on BlueField:

- Pre-run setup:

  a). The URL filter example is based on DPDK libraries. Therefore, the user is required to provide DPDK flags, and allocate huge pages.

  ```
  sudo echo 2048 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
  ```

  b). Make sure the RegEx engine is active:

  ```
  systemctl status mlx-regex
  ```

  If the status is inactive (`Active: failed`), run:

  ```
  systemctl start mlx-regex
  ```

- CLI example for running the app:

```
/opt/mellanox/doca/applications/url_filter/bin/doca_url_filter -a
 0000:03:00.0,class=regex -a auxiliary:mlx5_core.sf.4,sft_en=1 -a
 auxiliary:mlx5_core.sf.5,sft_en=1 -c3 -- -p
```

> **Note:** The SFT supports a maximum of 64 queues. Therefore, the application cannot be run with more than 64 cores. To limit the number of cores, run:
>
> ```
> /opt/mellanox/doca/applications/url_filter/bin/doca_url_filter -a
>  0000:03:00.0,class=regex -a auxiliary:mlx5_core.sf.4,sft_en=1 -a
>  auxiliary:mlx5_core.sf.5,sft_en=1 -l 0-64 -- -p
> ```
>
> This limits the application to 65 cores (core-0 to core-64) with 1 core for the main thread and 64 cores to serve as workers.

> **Note:** The flags `-a 0000:03:00.0,class=regex -a auxiliary:mlx5_core.sf.4,sft_en=1 -a auxiliary:mlx5_core.sf.5,sft_en=1` are necessary for proper usage of the application. Modifying them results in unexpected behavior as only 2 ports are supported. The subfunction number is arbitrary and configurable. The RegEx device, however, is not and must be initiated on port 0.

> **Note:** Sub-functions must be enabled according to the Scalable Function Setup Guide.

5. Running the application on the host, CLI example:

```
/opt/mellanox/doca/applications/url_filter/bin/doca_url_filter -a
 0000:05:00.0,class=regex  -a 05:00.3 -a 05:00.4 -v -- -p
```

> **Note:** Refer to section "Running DOCA Application on Host" in NVIDIA DOCA Virtual Functions User Guide.

6. a). To run `doca_url_filter` using a JSON file:

   ```
   doca_url_filter --json [json_file]
   ```

   For example:

   ```
   cd /opt/mellanox/doca/applications/url_filter/bin
   ./doca_url_filter --json url_params.json
   ```

URL Filter is based on user interaction with shell commands. Using the TAB key allows autocompletion while the `quit` command terminates the application. The following are other available commands:

- `create database` – removes and creates a new signature database at `/tmp/signature.txt` so it can be used in the filter command

- `filter http [msg] [regex]` – a signature containing the regular expression written is created in the database. When a match is found, a message is printed.
- `commit database [path]` – compiles and loads the signatures created by the filter command from the file path provided. The default database is `/tmp/signature.txt`.

**Note:** To load a signatures file that was created beforehand, simply run `commit database` with the desired path to load the file.

# Chapter 7. Arg Parser DOCA Flags

Refer to NVIDIA DOCA Arg Parser User Guide for more information.

| Flag Type | Short Flag | Long Flag/JSON Key | Description | JSON Content |
|---|---|---|---|---|
| DPDK Flags | a | devices | Add a PCI device into the list of devices to probe | `"devices": [ {"device": "regex",” id": "0000:03:00.0"}, {"device": "sf", "id": “4”,"sft": true}, {"device": "sf", "id": “5”,"sft": true}, ]` |
| | c | core-mask | Hexadecimal bitmask of cores to run on | `"core-mask": 3` |
| | l | core-list | List of cores to run on | `"core-list": "0-4"` |
| General Flags | l | log-level | Sets the log level for the application:<br>▶ CRITICAL=0<br>▶ ERROR=1<br>▶ WARNING=2<br>▶ INFO=3<br>▶ DEBUG=4 | `"log-level": 4` |
| | v | version | Print program version information | N/A |

| Flag Type | Short Flag | Long Flag/JSON Key | Description | JSON Content |
|---|---|---|---|---|
| | h | help | Print a help synopsis | N/A |
| Program Flags | p | print-match | Prints FID when matched in DPI engine | `"print-match": true` |
| | f | fragmented | Enables processing of fragmented packets | `"fragmented": false` |

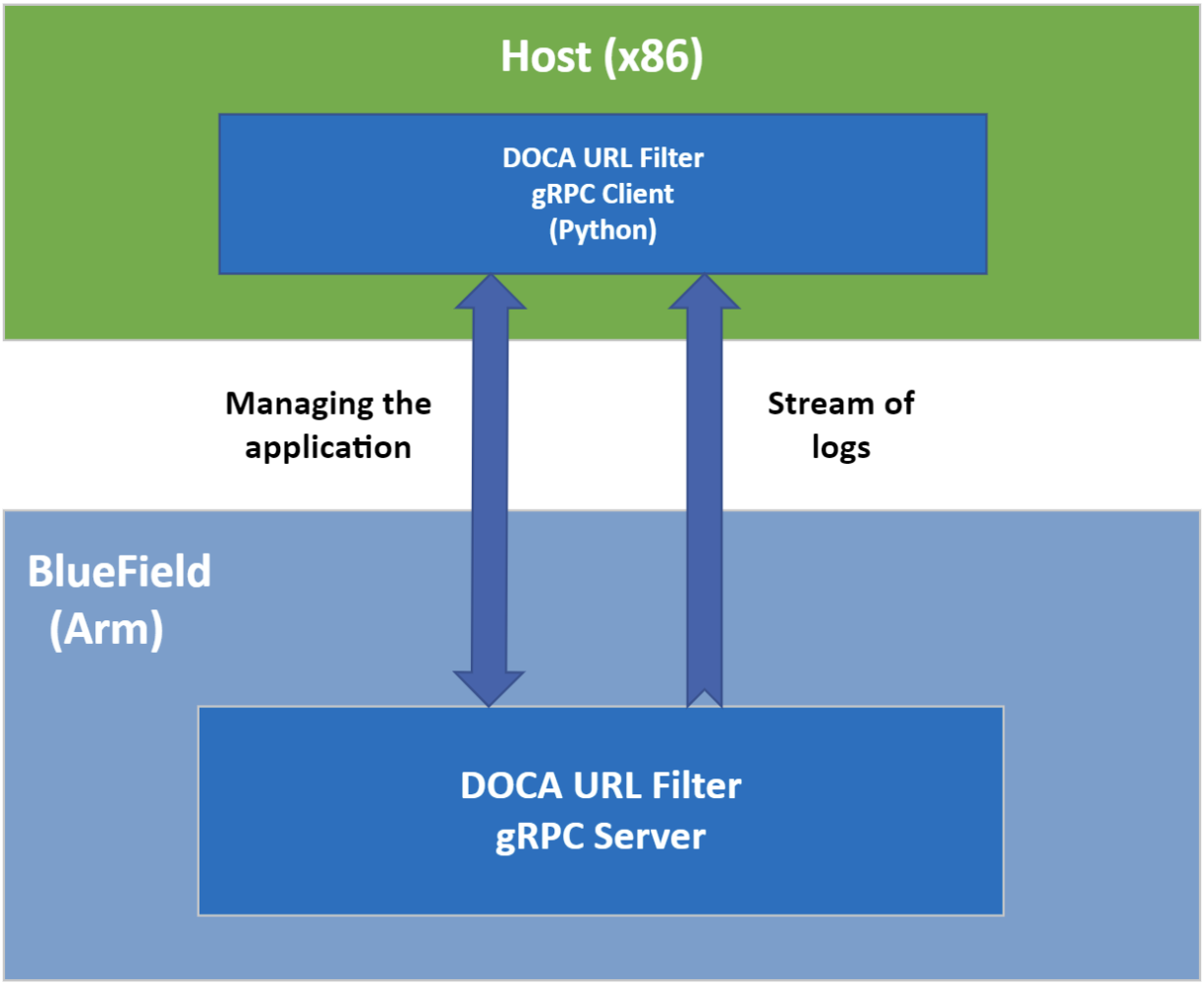
# Chapter 8. Deploying Containerized Application

The URL filter example supports a container-based deployment.

1. Refer to the NVIDIA DOCA Container Deployment Guide for details on how to deploy a DOCA container to the BlueField.
2. Notice that the container starts in sleeping mode. That is because of the interactive nature of the URL Filter example app. To run the example, attach to the running container. Run the following:
   ```
   sudo crictl exec -it <container-id> /bin/bash
   ```
   Then refer to the `entrypoint.sh` script at the root directory for an example of how to run the URL filter.
3. Application-specific configuration steps can be found on NGC, under the application's container page.

# Chapter 9. Managing gRPC-Enabled Application from Host

For instructions on running the gRPC application server on the BlueField, refer to NVIDIA DOCA gRPC Infrastructure User Guide.




To run the Python client of the gRPC-enabled application:

```
./doca_url_filter_gRPC_client.py -d/--debug <server address[:server port]>
```

For example:

```
/opt/mellanox/doca/applications/url_filter/bin/grpc/client/
doca_url_filter_gRPC_client.py -d 192.168.104.2
```

# Chapter 10. References

- /opt/mellanox/doca/applications/url_filter/src/url_filter.c
- /opt/mellanox/doca/applications/url_filter/src/grpc/url_filter.proto

## 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 nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume 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 Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright



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