



NVIDIA DOCA App Shield

Sample Guide

Table of Contents

Chapter 1. Introduction.....	1
Chapter 2. Dependencies.....	2
Chapter 3. Prerequisites.....	3
Chapter 4. Running the Sample.....	4
Chapter 5. Samples.....	5
5.1. Apsd Libs Get.....	5
5.2. Apsd Modules Get.....	5
5.3. Apsd Pslst.....	6
5.4. Apsd Threads Get.....	7
5.5. Apsd Vads Get.....	7
5.6. Apsd Envars Get.....	8
5.7. Apsd Privileges Get.....	9

Chapter 1. Introduction

DOCA App Shield is a library for monitoring the host and authenticating the integrity of core processes.

For more information about DOCA App Shield library, refer to [NVIDIA DOCA App Shield Programming Guide](#).

Chapter 2. Dependencies

The library requires a minimum DPU firmware version of 24.32.1010.

Chapter 3. Prerequisites

Make sure to follow the stages of the library prerequisites detailed in [NVIDIA DOCA App Shield Programming Guide](#) to make sure the library could be used by the samples. Afterwards, make sure to copy the generated JSON files into the DPU to the following path:

- ▶ `/tmp/symbols.json`
- ▶ `/tmp/mem_regions.json`

For more information regarding the runtime arguments, including how to get the VUID, please refer to section "Arg Parser DOCA Flags" of the [NVIDIA DOCA App Shield Agent Application Guide](#).

Chapter 4. Running the Sample

1. Refer to the following documents:

- ▶ [NVIDIA DOCA Installation Guide for Linux](#) 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. To build a given sample:

```
cd /opt/mellanox/doca/samples/doca_apsh/<sample_name>
meson build
ninja -C build
```



Note: The binary `doca_<sample_name>` will be created under `./build/`.

3. Sample (e.g., `apsh_libs_get`) usage:

```
Usage: doca_apsh_libs_get [DOCA Flags] [Program Flags]
```

DOCA Flags:

<code>-h, --help</code>	Print a help synopsis
<code>-v, --version</code>	Print program version information
<code>-l, --log-level</code>	Set the log level for the program
<code><CRITICAL=20, ERROR=30, WARNING=40, INFO=50, DEBUG=60></code>	

Program Flags:

<code>-p, --pid</code>	Process ID of process to be analyzed
<code>-f, --vuid</code>	VUID of the System device
<code>-d, --dma</code>	DMA device name
<code>-s, --osty <windows linux></code>	System OS type

For additional information per sample, use the `-h` option:

```
./build/doca_<sample_name> -h
```

Chapter 5. Samples

5.1. Apsh Libs Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of loadable libraries of a specific process.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCI device via given vendor unique identifier (VUID).
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of system process using Apsh API and searching for a specific process with the given PID.
8. Getting the list of process-loadable libraries using `doca_apsh_libs_get` Apsh API call.
9. Querying the libraries for 3 selected fields using `doca_apsh_lib_info_get` Apsh API call.
10. Printing libraries' attributes to the terminal.
11. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/apsh_libs_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/apsh_libs_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.2. Apsh Modules Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of installed modules on a monitored system.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCI device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and start Apsh system handler.
7. Getting the the list of system-installed modules using `doca_apsh_modules_get` Apsh API call.
8. Querying the names of modules using `doca_apsh_module_info_get` Apsh API call.
9. Printing the attributes of up to 5 moduless attributes to the terminal.
10. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/apsh_libs_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/apsh_libs_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_libs_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.3. Apsh Pslist

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of running processes on a monitored system.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCI device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of processes running on the system using `doca_apsh_processes_get` Apsh API call.
8. Querying the processes for 4 chosen attributes using `doca_apsh_proc_info_get` Apsh API call.
9. Printing the attributes of up to 5 processes to the terminal.
10. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_pslist/apsh_pslist_sample.c`

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_pslist/apsh_pslist_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_pslist/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.4. Apsh Threads Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of threads of a specific process.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCI device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of system processes using Apsh API and searching for a specific process with the given PID.
8. Getting the list of process threads using `doca_apsh_threads_get` Apsh API call.
9. Querying the threads for up to 3 selected fields using `doca_apsh_thread_info_get` Apsh API call.
10. Printing thread attributes to the terminal.
11. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_threads_get/apsh_threads_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_threads_get/apsh_threads_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_threads_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.5. Apsh Vads Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of virtual address descriptors (VADs) of a specific process.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.

3. Setting and start the Apsh context.
4. Opening DOCA remote PCI device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of system processes using Apsh API and searching for a specific process with the given PID.
8. Getting the list of process VADs using `doca_apsh_vads_get` Apsh API call.
9. Querying the VADs for 3 selected fields using `doca_apsh_vad_info_get` Apsh API call.
10. Printing the attributes of up to 5 VADs to the terminal.
11. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_vads_get/apsh_vads_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_vads_get/apsh_vads_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_vads_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.6. Apsh Envars Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of environment variables of a specific process.



Note: This sample works only on target systems with Windows OS.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCIe device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of system processes using Apsh API and searching for a specific process with the given PID.
8. Getting the list of process envvars using `doca_apsh_envvars_get` Apsh API call.
9. Querying the envvars for 2 selected fields using `doca_apsh_envvar_info_get` Apsh API call.
10. Printing the envvars attributes to the terminal.
11. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_envvars_get/apsh_envvars_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_envvars_get/apsh_envvars_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_envvars_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

5.7. Apsh Privileges Get

This sample illustrates how to properly initialize DOCA App Shield and use its API to get the list of privileges of a specific process.



Note: This sample works only on target systems with Windows OS.

The sample logic includes:

1. Opening DOCA device with DMA ability.
2. Creating DOCA Apsh context.
3. Setting and starting the Apsh context.
4. Opening DOCA remote PCIe device via given VUID.
5. Creating DOCA Apsh system handler.
6. Setting fields and starting Apsh system handler.
7. Getting the list of system processes using Apsh API and searching for a specific process with the given PID.
8. Getting the list of process privileges using the `doca_apsh_privileges_get` Apsh API call.
9. Querying the privileges for 5 selected fields using the `doca_apsh_privilege_info_get` Apsh API call.
10. Printing the privileges attributes to the terminal.
11. Cleaning up.

References:

- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_privileges_get/apsh_privileges_get_sample.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_privileges_get/apsh_privileges_get_main.c`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_privileges_get/meson.build`
- ▶ `/opt/mellanox/doca/samples/doca_apsh/apsh_common.c; /opt/mellanox/doca/samples/doca_apsh/apsh_common.h`

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 NON-INFRINGEMENT, 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

© 2022 NVIDIA Corporation & affiliates. All rights reserved.