NVIDIA DOCA Arg Parser

Programming Guide

MLNX-15-060520 _v2.2.0 | October 2023

Table of Contents

Chapter 1. Introduction 1
Chapter 2. API 2
2.1. doca_argp_param 2
2.2. doca_argp_param_create 2
2.3. doca_argp_register_param 2
2.4. doca_argp_set_dpdk_program 3
2.5. doca_argp_start 3
Chapter 3. DPDK Flags 4
Chapter 4. DOCA General Flags 6
Chapter 5. DOCA Program Flags 7
Chapter 6. JSON File Example 8

Chapter 1. Introduction

The Arg Parser module makes it simple to create a user command-line interface to supply program arguments. The module supports both regular command-line arguments and flags from a JSON file.

It also creates help and usage messages for all possible flags when the user provides invalid inputs to the program.

General notes about DOCA Arg Parser:

- Arg Parser checks a variety of errors including invalid arguments and invalid types, and it prints the error along with program usage and exits when it encounters an error
- The module uses long flags as JSON keys
- The options `-j` and `--json` are reserved for Arg Parser JSON and cannot be used

Chapter 2. API

For the library API reference, refer to ARGP API documentation in NVIDIA DOCA Libraries API Reference Manual.

Note: The pkg-config (`*.pc` file) for the Arg Parser library is included in DOCA's regular definitions (i.e., `doca`).

The following sections provide additional details about the library API.

2.1. doca_argp_param

The data structure contains the program flag details needed to process DOCA ARGP. These details are used to identify if the user inserts the flag in the command line, generate usage information for the flag, notify the program in case of matching, etc.

```
struct doca_argp_param;
```

2.2. doca_argp_param_create

Creates a DOCA ARGP parameter instance. The user is required to update the param attributes by calling the respective setter functions and registering the param by calling `doca_argp_register_param()`.

```
doca_error_t doca_argp_param_create(struct doca_argp_param **param);
```

Where:

- `param [out]` – DOCA ARGP param structure with unset attributes

2.3. doca_argp_register_param

Calling this function registers program flags in the Arg Parser database. The registration includes flag details. Those details are used to parse the input arguments and generate usage print. The user must register all program flags before calling `doca_argp_start()`.

```
doca_error_t doca_argp_register_param(struct doca_argp_param *input_param);
```

Where:

- `input_param [in]` – program flag details

2.4. doca_argp_set_dpdk_program

Marks the programs as a DPDK program. Once ARGP is finished with the parsing, DPDK (EAL) flags are forwarded to the program by calling the given callback function.

```
void doca_argp_set_dpdk_program(dpdk_callback callback);
```

Where:

- `callback [in]` – callback function to handle DPDK (EAL) flags

2.5. doca_argp_start

Calling this function starts the classification of command-line mode or JSON mode and is responsible for parsing DPDK flags if needed. If the program is triggered with a JSON file, the DPDK flags are parsed from the file and constructed in the correct format. DPDK flags are forwarded back to the program by calling the registered callback.

```
doca_error_t doca_argp_start(int argc, char **argv);
```

Where:

- `argc [in]` - number of input arguments
- `argv [in]` - program command-line arguments

Chapter 3. DPDK Flags

The following table lists the supported DPDK flags:

Short Flag	Long Flag/ JSON Key	Flag Description	JSON Content	JSON Content Description

a	devices	Add a PCIe device to the list of devices to probe	<pre>"devices": [
			{
			"device": "regex",
			"id": "03:00.0"
			},
			{
			"device": "sf",
			"id": "4",
			"sft": true
			},
			{
			"device": "sf",
			"id": "5",
			"hws": true
			},
			{
			"device": "vf",
			"id": "b1:00.3"
			},
			{
			"device": "pf",
			"id": "03:00.0",
			"sft": true
			},
			{
			"device": "gpu",
			"id": "06:00.0"
			}</pre>
				Passing configuration for six devices:
				1. RegEx device with PCIe address 03:00.0.
				2. SF device with number 4, SFT enabled.
				3. SF device with number 5, HW steering enabled.
				4. VF device with PCIe b1:00.3.
				5. PF device with PCIe address 03:00.0, SFT enabled.
				6. GPU device with PCIe address 06:00.0.

Short Flag	Long Flag/ JSON Key	Flag Description	JSON Content	JSON Content Description
			]	
c	core-mask	Hexadecimal bitmask of cores to run on	"core-mask": "0xff"	Set core mask with value 0xff
l	core-list	List of cores to run on	"core-list": "0-4"	Limit program to use five cores (core-0 to core-4)

Note: Additional DPDK flags may be added in the "flags" JSON field.

Chapter 4. DOCA General Flags

The following table lists the supported DOCA general flags:

Short Flag	Long Flag/ JSON Key	Flag Description	JSON Content	JSON Content Description
l	log-level	Sets the log level for the application: ▶ CRITICAL=20 ▶ ERROR=30 ▶ WARNING=40 ▶ INFO=50 ▶ DEBUG=60	`"log-level": 60`	Set the log level to DEBUG mode
v	version	Print program version information	N/A	Supported only on CLI
h	help	Print a help synopsis	N/A	Supported only on CLI
g	grpc-address	Set the IP address for the gRPC server Note: This flag is mandatory for gRPC mode.	`"grpc-address": "10.10.10.10"`	Set the server's IP address to 10.10.10.10
			`"grpc-address": "10.10.10.10:1234"`	Set the server's IP address to 10.10.10.10 and the server's port to 1234

Chapter 5. DOCA Program Flags

The flags for each program can be found in the document dedicated to that program, including instructions on how to run it, whether by providing a JSON file or by using the command-line interface.

Chapter 6. JSON File Example

An application JSON file can be found under `/opt/mellanox/applications/[APP name]/bin/[APP name]_params.json`.

```
{
  "doca_dpdk_flags": {
    // -a - Add a device to the allow list.
    "devices": [
      {
        "device": "sf",
        "id": "4",
        "sft": true
      },
      {
        "device": "sf",
        "id": "5",
        "sft": true
      }
    ],
    // -c - Hexadecimal bitmask of cores to run on
    "core-mask": "0xff",
    // Additional DPDK (EAL) flags (if needed)
    "flags": ""
  },
  // flags below are for URL Filter application.
  "doca_general_flags": {
    // -l - sets the log level for the application <CRITICAL=20, ERROR=30,
 WARNING=40, INFO=50, DEBUG=60>
    "log-level": 60
  },
  // flags below are for URL Filter application.
  "doca_program_flags": {
    // -p - prints FID when matched in DPI engine
    "print-match": true,

    // -a - set the regex engine pci address
    "pci-addr": "03:00.0"
  }}
```

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

© 2023 NVIDIA Corporation & affiliates. All rights reserved.

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