NVIDIA DOCA IPsec

Programming Guide

MLNX-15-060568 _v2.2.1 | October 2023

Table of Contents

Chapter 1. Introduction

DOCA IPsec provides an API to create the security association (SA) object required for flow encryption and decryption hardware acceleration.

Using DOCA IPsec can accelerate various IPsec actions in hardware such as:

- IPsec Crypto Offload – accelerate ciphering (encryption and decryption) only
- IPsec Full Offload – accelerate all IPsec actions (i.e., encap/decap, ciphering, and replay protection so sequence numbers are done in the NIC hardware)
- Partial – doing some IPsec actions in hardware and others in software
- Transparent IPsec – a variant of IPsec full offload, data-path endpoint is not aware of IPsec (only available in BlueField)

For more information about flow encryption and decryption, please refer to the NVIDIA DOCA Flow Programming Guide.

This document is intended for software developers wishing to accelerate their application's flow encryption and decryption operations.

Chapter 2. Prerequisites

DOCA IPsec-based applications can run either on the host machine or on the NVIDIA® BlueField® DPU target.

Confirm that your BlueField DPU is crypto-enabled:

```
mlxfwmanager | grep Crypto # expected "Crypto Enabled;"
```

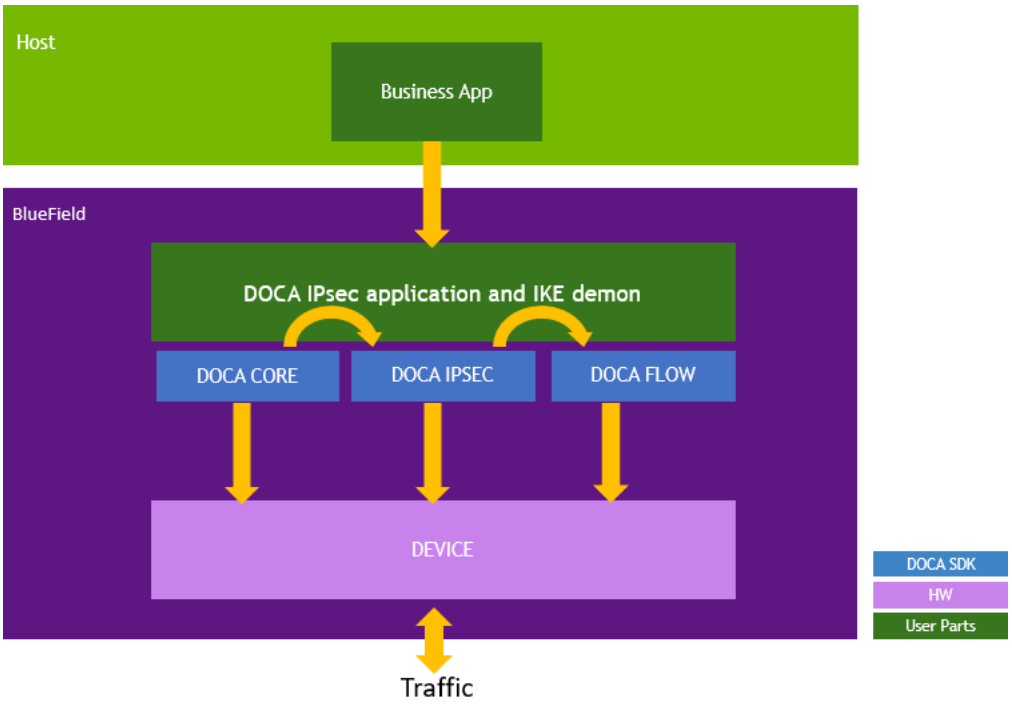
Note: Make sure IPsec is supported using the `doca_ipsec_job_get_supported` function. IPsec is supported in the DPU or on host when DPU is in NIC mode only.

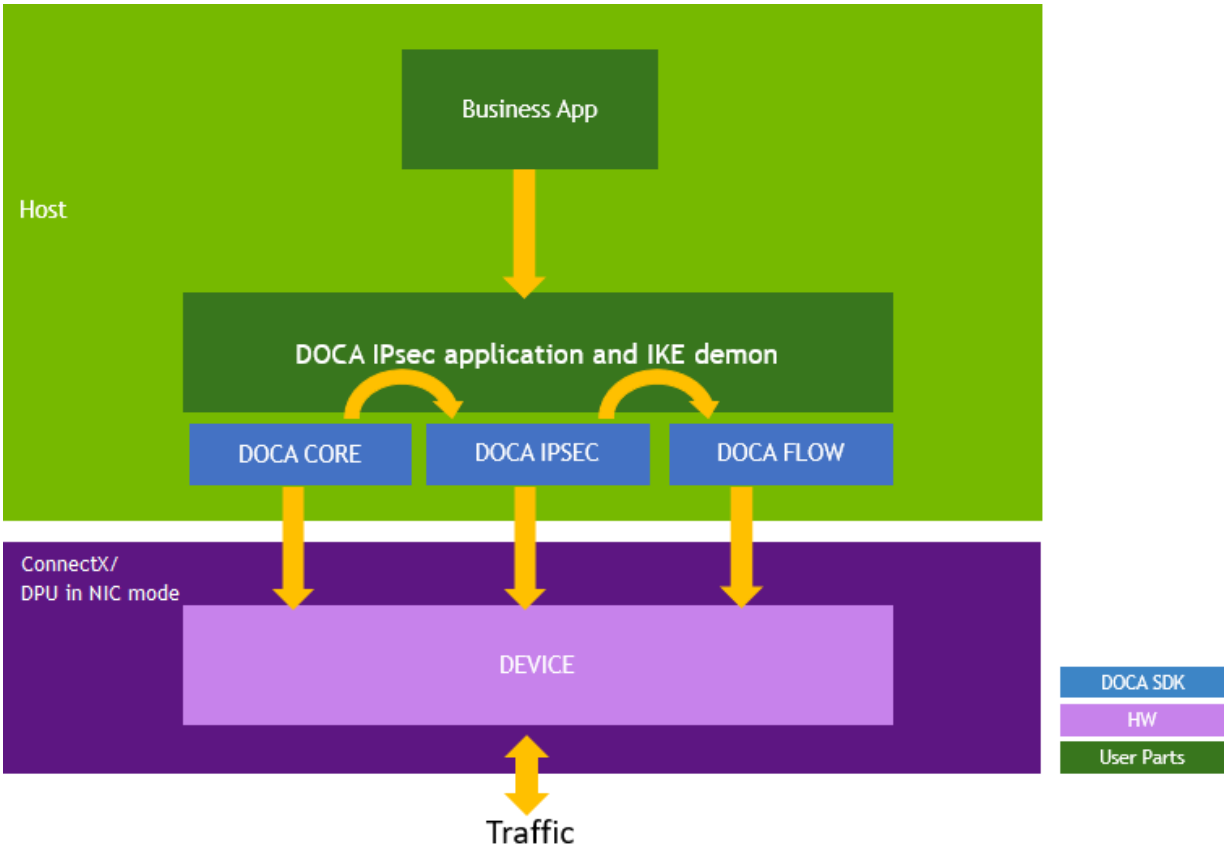
If a hardware sequence number or hardware antireplay are used, IPsec support is enabled using `doca_ipsec_sequence_number_get_supported` and `doca_ipsec_antireplay_get_supported` respectively.

Chapter 3. Architecture

DOCA IPsec relies heavily on the underlying DOCA core architecture for its operation.

After initialization, a DOCA IPsec operation is requested by submitting an IPsec job on the relevant work queue. The DOCA IPsec library then calls a progress retrieve action to post a completion event on the work queue.




Business App
Host
DOCA IPsec application and IKE demon
DOCA CORE
DOCA IPSEC
DOCA FLOW
ConnectX/
DPU in NIC mode
DEVICE
Traffic
DOCA SDK
HW
User Parts

Chapter 4. API

This section details the specific structures and operations related to the DOCA IPsec library for general initialization, setup, and clean-up.

The API for DOCA IPsec consists of the main DOCA IPsec job structure that is passed to the work queue to instruct the library on source attributes and SA output.

As with other libraries, the DOCA IPsec job contains the standard `doca_job` base field that should be set as follows:

- To create a job:

```
/* Construct IPsec job */
{
    .base = (struct doca_job){
        .type = DOCA_IPSEC_JOB_SA_CREATE,
        .flags = DOCA_JOB_FLAGS_NONE,
        .ctx = state.ctx
      },
      .sa_attrs = sa_attrs,
};
```

SA attributes:

```
struct doca_ipsec_sa_attrs {
 struct doca_encryption_key key;                  /**< IPSec encryption key */
 enum doca_ipsec_icv_length icv_length;           /**< Authentication Tag length
 */
 struct doca_ipsec_sa_attr_sn sn_attr;            /**< sn attributes */
 enum doca_ipsec_direction direction;             /**< egress/ingress */
 union {                                          /**< egress/ingress attr */
           struct doca_ipsec_sa_attr_egress egress;     /**< egress attr */
           struct doca_ipsec_sa_attr_ingress ingress;  /**< ingress attr */
    };
    struct doca_ipsec_sa_event_attrs event;             /**< Reserve future use -
 ipsec events flags */
};

struct doca_ipsec_sa_attr_sn {
 uint32_t esn_overlap; /**< new/old indication of the High sequence number MSB -
 when set is old */
 uint32_t esn_enable;  /**< when set esn is enabled */
 uint64_t sn_initial;  /**< set the initial sequence number */
};

struct doca_ipsec_sa_attr_egress {
    uint32_t sn_inc_enable; /**< when set sn increment offloaded */
};

struct doca_ipsec_sa_attr_ingress {
    uint32_t antireplay_enable;
    /**< when enabled activates anti-replay protection window. */
    enum doca_ipsec_replay_win_size replay_win_sz;
```

```
    /**< Anti replay window size to enable sequence replay attack handling. */
};


struct doca_ipsec_sa_event_attrs {
    uint32_t remove_flow_packet_count;
    /**< Packet counter, Decrements for every packet passing through the SA.
     * Event are triggered occurs when the counter reaches soft- lifetime and
 hard-lifetime (0).
     * When counter reaches hard-lifetime, all passing packets will return a
 relevant Syndrome.
     */
    uint32_t remove_flow_soft_lifetime;
    /**< Soft Lifetime threshold value.
     * When remove_flow_packet_count reaches this value a soft lifetime event is
 triggered (if armed).
     * See remove_flow_packet_count field in this struct fro more details.
     */
 uint32_t soft_lifetime_arm;
 /**< 1 when armed/to arm 0 otherwise. */
 uint32_t hard_lifetime_arm;
 /**< 1 when armed/to arm 0 otherwise. */
 uint32_t remove_flow_enable;
 /**< 1 when remove flow enabled/to enable; 0 otherwise. */
 uint32_t esn_overlap_event_arm;
 /**< 1 when armed/to arm 0 otherwise. */
};
```

As with all WorkQ operations, the application must periodically poll the work queue (via `doca_workq_progress_retrieve` API call). When the retrieve call returns with a pointer to an SA object value (to indicate that the work queues event is valid), you can then test that received event for success:

```
struct doca_ipsec_sa* sa = doca_ipsec_sa_from_result(&event);
```

- To destroy a job:

  As with all WorkQ operations, the application must periodically poll the work queue (via `doca_workq_progress_retrieve` API call). When the retrieve call returns with a `DOCA_SUCCESS` value (to indicate the work queues event is valid), you can then test that received event for success:

```
event.result.u64 == DOCA_SUCCESS
```

Chapter 5. Usage

The following step-by-step guide goes through the various stages required to initialize, execute, and clean-up DOCA IPsec API.

5.1. Initialization Process

The DOCA IPsec API uses the DOCA core library to create the required objects for the DOCA IPsec library operations. This section runs through this process in a logical order. If you already have some of these operations in your DOCA application, you may skip or modify them as needed.

5.1.1. Opening DOCA Device

The first requirement is to open a DOCA device, normally your BlueField controller. You should iterate all DOCA devices (via `doca_devinfo_list_create`) and select one using some criteria (e.g., PCIe address). You may also use the function `doca_ipsec_job_get_supported` to check if the device is suitable for the DOCA IPsec job type you want to perform. Afterwards, the device should be opened using `doca_dev_open`.

Note: If a hardware sequence number or hardware antireplay are used, IPsec support is enabled using `doca_ipsec_sequence_number_get_supported` and `doca_ipsec_antireplay_get_supported` respectively.

5.1.2. Creating DOCA Core Objects

DOCA IPsec also requires the actual DOCA IPsec context to be created (`doca_ipsec_create`).

5.1.3. Initializing DOCA Core Objects

In this phase of initialization, the core objects are ready to be set up and started.

5.1.3.1. DOCA IPsec Context Initialization

The context created previously can have the device added (`doca_ctx_dev_add`), started (`doca_ctx_start`), and work queue added (`doca_ctx_workq_add`).

5.1.4. Constructing DOCA IPsec Attributes

Prior to building and submitting a DOCA IPsec operation, you must construct DOCA IPsec attributes.

5.2. IPsec Execution

The DOCA IPsec operation works with the DOCA core work queue. Therefore, you must enqueue the operation and then poll for completion.

5.2.1. Constructing and Executing DOCA IPsec Operation

To begin the DOCA IPsec operation, you must enqueue a DOCA IPsec job on the previously created work queue object. This involves creating the DOCA IPsec job (`struct doca_ipsec_sa_create_job`) that is a composite of specific DOCA IPsec fields.

Finally, the `doca_workq_submit` API call is used to submit the DOCA IPsec operation to the work queue.

5.2.2. Waiting for Completion

To detect when the DOCA IPsec operation has completed, you should periodically poll the work queue (via `doca_workq_progress_retrieve`).

When the API call indicates that a valid event has been received, you should detect the success of the DOCA IPsec operation through the `event.result.u64` field which would be a pointer to the SA object upon creation, or `DOCA_SUCCESS` upon destruction. It should be noted that other work queue operations (i.e., non-DOCA IPsec operations) present their events differently. Refer to their respective guides for more information.

Upon completion, convert the event to an SA object (`doca_ipsec_sa_from_result`). DOCA Flow requires the SA object for encryption and decryption.

To clean up the SA object, use the destroy SA job with the SA pointer.

5.2.3. Clean-up

The main clean-up process is to remove the worker queue from the context (`doca_ctx_workq_rm`), stop the context itself (`doca_ctx_stop`), and remove the device from the context (`doca_ctx_dev_rm`).

The final destruction of the objects can now occur. This can occur in any order, but destruction must occur on the work queue (`doca_workq_destroy`), IPsec context (`doca_ipsec_destroy`), and device closure (`doca_dev_close`).

Note: Destroying SA objects results in an error upon device closure.

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