NVIDIA DOCA IPsec

Programming Guide
Table of Contents

Chapter 1. Introduction........................................................................................................ 1
Chapter 2. Prerequisites...................................................................................................... 2
Chapter 3. Architecture........................................................................................................ 3
Chapter 4. API.......................................................................................................................4
Chapter 5. Usage.................................................................................................................. 6
  5.1. Initialization Process.................................................................................................................6
    5.1.1. Opening DOCA Device ........................................................................................................ 6
    5.1.2. Creating DOCA Core Objects .............................................................................................6
    5.1.3. Initializing DOCA Core Objects .......................................................................................... 6
      5.1.3.1. DOCA IPsec Context Initialization ...............................................................................6
    5.1.4. Constructing DOCA IPsec Attributes .................................................................................6
  5.2. IPsec Execution.........................................................................................................................7
    5.2.1. Constructing and Executing DOCA IPsec Operation ........................................................ 7
    5.2.2. Waiting for Completion ................................................................................................... 7
    5.2.3. Clean-up ............................................................................................................................7
Chapter 1. Introduction

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

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.
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.
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:**

  ```c
  /* 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:

```c
struct doca_ipsec_sa_attrs {
  enum doca_ipsec_sa_mode mode;               /**< IPsec protocol
                                             - transport of tunnel */
  enum doca_ipsec_sa_offload offload;         /**< offload type - full or only crypto - only supported DOCA_IPSEC_SA_OFFLOAD_FULL; */
  enum doca_ipsec_sa_protocol protocol;       /**< protocol type - esp or ah - only supported DOCA_IPSEC_SA_PROTO_ESP */
  uint32_t direction;                         /**< ingress/encrypt */
  uint32_t esn_enabled;                       /**< when set esn is enabled */
  uint32_t esn_overlap;                       /**< new/old indication of the High sequence number MSB - when set is old */
  enum doca_ipsec_icv_length icv_length;      /**< Authentication Tag length */
  uint32_t spi;                               /**< SA security parameter index */
  struct doca_encryption_key key;             /**< IPsec encryption key */
  struct doca_ipsec_sa_antireplay antireplay; /**< IPsec antireplay attr */
};
```

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:

```c
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:

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

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.