DOCA LIBRARIES API

Reference Manual

Table of Contents

Chapter 1. Change Log

This chapter list changes in API that were introduced to the library.

1.3.0

▶ Field Groups, GPU Groups, and field watches created with a handle returned from dcgmConnect() are now cleaned up upon disconnect. dcgmConnect_v2() can be used to get the old behavior of objects persisting after disconnect.

▶ dcgmConnect_v2() was added as a method for specifying additional connection options when connecting to the host engine.

▶ dcgmUnwatchFields() was added as a method of unwatching fields that were previously watched with dcgmWatchFields()

▶ dcgmActionValidate_v2() was added to be able to pass more parameters to the DCGM GPU Diagnostic.

▶ dcgmDiagResponse_t was increased from v2 to v3. See dcgmDiagResponse_v3 for details

1.2.3

▶ No API changes in this version.

1.1.1

▶ dcgmGetAllSupportedDevices() was added as a method to get DCGM-supported GPU Ids. dcgmGetAllDevices() can still be used to get all GPU Ids in the system.

1.0.0

▶ Initial Release.

Chapter 2.   Modules

Here is a list of all modules:

- App Shield
- Arg Parser
- Core

  - Buffer
  - Buffer Inventory
  - Compatibility Management
  - DOCA Context
  - DOCA Device
  - DOCA Error
  - Logging Management
  - Memory Map
  - Version Management
- Comm Channel
- DOCA DMA engine
- Deep packet inspection
- Remote deep packet inspection (grpc)
- Flow
- Flow (grpc)
- Flow net
- RegEx engine
- RegEx engine memory pool
- Telemetry Service Library

2.1. App Shield

DOCA App Shield library let you to monitor operation system that resides on the host. This is done with the DPU DMA capabilities and the regex engine. Please follow the programmer guide for system configurations.

enum doca_apsh_system_layer

system supported layer types

Values

DOCA_APSH_LAYER_BARE_METAL
Bare metal system - no abstraction layer
DOCA_APSH_LAYER_VM
Virtual system
DOCA_APSH_LAYER_DOCKER_CONTAINER
Docker process

enum doca_apsh_system_os

system os types

Values

DOCA_APSH_SYSTEM_LINUX
linux
DOCA_APSH_SYSTEM_WINDOWS
windows

const __DOCA_EXPERIMENTAL void *__doca_apsh_attst_info_get (doca_apsh_attestation *attestation, enum doca_apsh_attestation_attr attr)

Shadow function - get attribute value for a attestation.

Parameters

attestation
single attestation handler
attr
Attribute to get the info on the attestation

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_attestation_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_lib_info_get (doca_apsh_lib *lib, enum doca_apsh_lib_attr attr)

Shadow function - get attribute value for a lib.

Parameters

lib
single lib handler
attr
Attribute to get the info on the lib

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_lib_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_module_info_get (doca_apsh_module *module, enum doca_apsh_module_attr attr)

Shadow function - get attribute value for a module.

Parameters

module
single module handler
attr
Attribute to get the info on the module

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_mod_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_proc_info_get (doca_apsh_process *process, enum doca_apsh_process_attr attr)

Shadow function - get attribute value for a process.

Parameters

process
    single process handler
attr
    Attribute to get the info on the process

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_proc_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_thread_info_get (doca_apsh_thread *thread, enum doca_apsh_thread_attr attr)

Shadow function - get attribute value for a thread.

Parameters

thread
    single thread handler
attr
    Attribute to get the info on the thread

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_thread_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_vad_info_get (doca_apsh_vad *vad, enum doca_apsh_vad_attr attr)

Shadow function - get attribute value for a vad.

Parameters

vad
    single vad handler
attr
    Attribute to get the info on the vad

Returns

return the info requested, need to cast

Description

Do not use this function, recommanded to use doca_apsh_vad_info_get

__DOCA_EXPERIMENTAL void doca_apsh_attestation_free (doca_apsh_attestation **attestation)

Destroys a attestation context.

Parameters

attestation
    Attestation opaque pointer of the process to destroy

doca_error_t doca_apsh_attestation_get (doca_apsh_process *process, const char *exec_hash_map_path, doca_apsh_attestationattestation, int *attestation_size)

Get current process attestation.

Parameters

process
 Process handler
exec_hash_map_path
 path to file containing the hash calculations of the executable and dlls/libs of the process note that changing the process code or any libs can effect this. The file can be created by running the doca_exec_hash_build_map tool on the system.
attestation
 Attestation opaque pointers of the process
attestation_size
 Output param, will contain size of attestation array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_INITIALIZATION - if modules list initialization failed.
- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.
- ▶ DOCA_ERROR_NOT_FOUND - if process hasn't been found.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return is snapshot, this is not dynamic, need to free it.

doca_error_t doca_apsh_attst_refresh (doca_apsh_attestationattestation, int *attestation_size)

refresh single attestation handler of a process with new snapshot

Parameters

attestation
    single attestation handler to refresh
attestation_size
    Output param, will contain size of attestation array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_INITIALIZATION - if modules list initialization failed.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.

▶ DOCA_ERROR_NOT_FOUND - if process hasn't been found.

Description

This function is multithreaded compatible with diffrent system context, Refresh the snapshot of the handler. Recommended to query all wanted information before refreshing.

__DOCA_EXPERIMENTAL doca_apsh_ctx *doca_apsh_create (void)

Create a new apsh handler.

Returns

apsh context required for creating system handler, NULL on failure

Description

Allocate memory and init the opaque struct for apsh handler. Before using the system handler use doca_apsh_start

__DOCA_EXPERIMENTAL void doca_apsh_destroy (doca_apsh_ctx *ctx)

Free the APSH memory and close connections.

Parameters

ctx

apsh context to destroy

doca_error_t doca_apsh_dma_dev_set (doca_apsh_ctx *ctx, doca_dev *dma_dev)

Set apsh dma device.

Parameters

ctx

apsh handler

dma_dev

doca device with dma capabilities, please reffer to doca_dev.h

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for dma_dev_name.

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL void doca_apsh_libs_free (doca_apsh_lib **libs)

Destroys a libs context.

Parameters

libs

Array of libs opaque pointers of the process to destroy

doca_error_t doca_apsh_libs_get (doca_apsh_process *process, doca_apsh_liblibs, int *libs_size)

Get array of current process loadable libraries.

Parameters

process
   Process handler
libs
   Array of libs opaque pointers of the process
libs_size
   Output param, will contain size of libs array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶   DOCA_ERROR_INITIALIZATION - if libs list initialization failed.

▶   DOCA_ERROR_NO_MEMORY - if cannot alloc memory to libs array.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

__DOCA_EXPERIMENTAL void doca_apsh_module_free (doca_apsh_module **modules)

Destroys a modules array.

Parameters

modules
   Array of module opaque pointers of the systems to destroy

doca_error_t doca_apsh_modules_get (doca_apsh_system *system, doca_apsh_modulemodules, int *modules_size)

Get array of current modules installed on the system.

Parameters

system
System handler

modules
Array of module opaque pointers of the systems

modules_size
Output param, will contain size of modules array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_INITIALIZATION - if modules list initialization failed.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

__DOCA_EXPERIMENTAL void doca_apsh_processes_free (doca_apsh_process **processes)

Destroys a process context.

Parameters

processes
Array of process opaque pointers of the systems to destroy

doca_error_t doca_apsh_processes_get (doca_apsh_system *system, doca_apsh_processprocesses, int *processes_size)

Get array of current processes running on the system.

Parameters

system
System handler
processes
Array of process opaque pointers of the systems
processes_size
Output param, will contain size of processes array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶   DOCA_ERROR_INITIALIZATION - if processes list initialization failed.

▶   DOCA_ERROR_NO_MEMORY - if cannot alloc memory to processes array.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

doca_error_t doca_apsh_regex_dev_set (doca_apsh_ctx *ctx, const char *regex_dev_name)

Set apsh regex device.

Parameters

ctx
apsh handler
regex_dev_name
device name with the capabilities of regex

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for regex_dev_name.

Description

This is a Mandatory setter

doca_error_t doca_apsh_start (doca_apsh_ctx *ctx)

Start apsh handler.

Parameters

ctx
    App Shield handler

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Start apsh handler and init connection to devices. Need to set apsh params with setter functions before starting the system. Mandatory setters: doca_apsh_dma_dev_set. Other setters can be query automatically but will take time.

doca_error_t doca_apsh_sys_dev_set (doca_apsh_system *system, doca_dev_remote *dev)

Set system device.

Parameters

system
    system handler
dev
    the device that is connected to the system to be queried. for example a vf that is connected to a vm or pf that is connected to the baremetal. doca remote device from dma device configed in doca_apsh_dma_dev_set. to query the right device please refer to doca_dev.h for full options.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This is a Mandatory setter

doca_error_t doca_apsh_sys_mem_region_set (doca_apsh_system *system, const char *system_mem_region_path)

Set system allowed memory regions.

Parameters

system
    system handler
system_mem_region_path
    path to json file containing the memory regions of the devices The memory regions are uniq per system, would not change on reboot or between different devices of the same system. note that adding/removing device from the host can change the regions. The json can be created by running the doca_system_mem_region tool on the system.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
▶   DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.

Description

This is a Mandatory setter

doca_error_t doca_apsh_sys_os_symbol_map_set (doca_apsh_system *system, const char *system_os_symbol_map_path)

Set system os symbol map.

Parameters

system
    system handler

system_os_symbol_map_path

the os memory map data, uniq per os build please note that changing linux kernel (adding/removing modules) will change the map should be created by running the doca_system_os_symbol_map tool on the system os

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.

Description

This is a Mandatory setter

doca_error_t doca_apsh_sys_os_type_set (doca_apsh_system *system, doca_apsh_system_os os_type)

Set system os type.

Parameters

system

system handler

os_type

system os type - windows/linux

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type had been received.

Description

This is a must setter

doca_error_t doca_apsh_sys_system_layer_set (doca_apsh_system *system, doca_apsh_system_layer layer_type)

Set system layer type.

Parameters

system
   system handler
layer_type
   system layer type - bare metal/vm ...

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This is a optional setter

__DOCA_EXPERIMENTAL doca_apsh_system *doca_apsh_system_create (doca_apsh_ctx *ctx)

Create a new system handler.

Parameters

ctx
   apsh handler

Returns

returns system pointer, NULL on failure

Description

Allocate memory and init the opaque struct for system handler. Before using the system handler use doca_apsh_system_start

__DOCA_EXPERIMENTAL void doca_apsh_system_destroy (doca_apsh_system *system)

Destroy system handler.

Parameters

system

system context to destroy

Description

This will not destroy process/module/libs ...

doca_error_t doca_apsh_system_start (doca_apsh_system *system)

Start system handler.

Parameters

system

system handler

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if app-shield system initialization have failed.

Description

Start system handler and init connection to the system. Need to set system params with setter functions before starting the system. Mandatory setters: os_symbol_map, mem_region, dev. Other setters can be query automatically but will take time.

__DOCA_EXPERIMENTAL void doca_apsh_threads_free (doca_apsh_thread **threads)

Destroys a threads context.

Parameters

threads

Array of threads opaque pointers of the process to destroy

doca_error_t doca_apsh_threads_get (doca_apsh_process *process, doca_apsh_threadthreads, int *threads_size)

Get array of current process threads.

Parameters

process

Process handler

threads

Array of threads opaque pointers of the process

threads_size

Output param, will contain size of threads array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_INITIALIZATION - if threads list initialization failed.

▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to threads array.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

__DOCA_EXPERIMENTAL void doca_apsh_vads_free (doca_apsh_vad **vads)

Destroys a vads context.

Parameters

vads

Array of vads opaque pointers of the process to destroy

doca_error_t doca_apsh_vads_get (doca_apsh_process *process, doca_apsh_vadvads, int *vads_size)

Get array of current process vads - virtual address descriptor.

Parameters

process

Process handler

vads

Array of vads opaque pointers of the process

vads_size

Output param, will contain size of vads array on success.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶   DOCA_ERROR_INITIALIZATION - if modules list initialization failed.

▶   DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.

▶   DOCA_ERROR_NOT_FOUND - if process hasn't been found.

Description

This function is multithreaded compatible with diffrent system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

#define doca_apsh_attst_info_get ((attr##_TYPE)(uintptr_t)__doca_apsh_attst_info_get(attestation, attr))

Get attribute value for a attestation.

Get the requested info from attestation handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_lib_info_get ((attr##_TYPE)(uintptr_t)__doca_apsh_lib_info_get(lib, attr))

Get attribute value for a lib.

Get the requested info from lib handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_module_info_get ((attr##_TYPE)(uintptr_t)__doca_apsh_module_info_get(module, attr))

Get attribute value for a module.

Get the requested info from module handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_proc_info_get ((attr##_TYPE)(uintptr_t)__doca_apsh_proc_info_get(process, attr))

Get attribute value for a process.

Get the requested info from process handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_thread_info_get ((attr##_TYPE)__doca_apsh_thread_info_get(thread, attr))

Get attribute value for a thread.

Get the requested info from thread handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_vad_info_get ((attr##_TYPE)(uintptr_t)__doca_apsh_vad_info_get(vad, attr))

Get attribute value for a vad.

Get the requested info from vad handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

2.2.   Arg Parser

DOCA Arg Parser library. For more details please refer to the user guide on DOCA devzone.

struct doca_argp_param

Program flag information.

struct doca_argp_program_general_config

DOCA general flags values as provided to the program.

struct doca_argp_program_type_config

Information about program configuration.

enum doca_argp_type

Flag input type.

Values

DOCA_ARGP_TYPE_STRING = 0
 Input type is a string
DOCA_ARGP_TYPE_INT
 Input type is an integer
DOCA_ARGP_TYPE_BOOLEAN

Input type is a boolean

DOCA_ARGP_TYPE_JSON_OBJ

DPDK Param input type is a json object, only for json mode

typedef (*callback_func) (void* , void*)

Flag callback function type.

__DOCA_EXPERIMENTAL void doca_argp_destroy (void)

ARG Parser destroy.

Description

cleanup all resources include calling rte_eal_cleanup(), to release EAL resources that has allocated during rte_eal_init().

> 📝 **Note:**
>
> After this call, no DPDK function calls may be made.

__DOCA_EXPERIMENTAL void doca_argp_init (const char *program_name, doca_argp_program_type_config *type_config, void *program_config)

Initialize the parser interface.

Parameters

program_name

Name of current program, using the name for usage print.

type_config

Announce if current program is based on DPDK/gRPC API.

program_config

Program configuration struct.

__DOCA_EXPERIMENTAL void doca_argp_register_param (doca_argp_param *input_param)

Register a program flag.

Parameters

input_param
    Program flag details.

Description

> 📵 **Note:**
>
> Value of is_cli_only field may be changed in this function.

__DOCA_EXPERIMENTAL void doca_argp_register_validation_callback (callback_func callback)

Register program validation callback function.

Parameters

callback
    Program validation callback.

__DOCA_EXPERIMENTAL void doca_argp_register_version_callback (callback_func callback)

Register an alternative version callback.

Parameters

callback
    Program-specific version callback.

__DOCA_EXPERIMENTAL void doca_argp_start (int argc, char **argv, doca_argp_program_general_config **general_config)

Parse incoming arguments (cmd line/json).

Parameters

argc
   Number of program command line arguments.
argv
   Program command line arguments.
general_config
   DOCA wide input arguments (log_level, ...).

Description

> **Note:**
>
> : if the program is DPDK app, doca_argp_start() will parses DPDK flags and calling rte_eal_init().

__DOCA_EXPERIMENTAL void doca_argp_usage (void)

Print usage instructions and exit with failure.

#define MAX_SERVER_ADDRESS 24

Maximum length for gRPC serever address string.

2.3. Core

Buffer

Buffer Inventory

Compatibility Management

DOCA Context

DOCA Device

DOCA Error

Logging Management

Memory Map

Version Management

2.3.1. Buffer

Core

The DOCA Buffer is used for reference data. It holds the information on a memory region that belongs to a DOCA memory map, and its descriptor is allocated from DOCA Buffer Inventory. Among other functions, it can be used to perform DMA operations.

doca_error_t doca_buf_head_get (doca_buf *buf, uint8_t **head)

Get the payload buffer pointed by the object.

Parameters

buf
    DOCA Buf element.
head
    The address of the payload buffer.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_len_get (doca_buf *buf, size_t *len)

Get the length of the payload buffer pointed by the object.

Parameters

buf

DOCA Buf element.

len

The length of the payload buffer pointed to by buf.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_refcount_add (doca_buf *buf, uint16_t *refcount)

Increase the object reference count by 1.

Parameters

buf

DOCA Buf element.

refcount

The number of references to the object before this operation took place.

Returns

▶ DOCA_ERROR_NOT_SUPPORTED

Description

📣 **Note:**

This function is not supported yet.

doca_error_t doca_buf_refcount_get (doca_buf *buf, uint16_t *refcount)

Get the reference count of the object.

Parameters

buf
　　DOCA Buf element.
refcount
　　The number of references to the object.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▸　DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_refcount_rm (doca_buf *buf, uint16_t *refcount)

Decrease the object reference count by 1, if 0 reached, return the element back to the inventory.

Parameters

buf
　　DOCA Buf element.
refcount
　　The number of references to the object before this operation took place.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▸　DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

When refcont 0 reached, all related resources should be released. For example if the element points into some mmap its state will be adjusted accordingly.

2.3.2.　Buffer Inventory

Core

The DOCA buffer inventory manages a pool of doca_buf objects. Each buffer obtained from an inventory is a descriptor that points to a memory region from a doca_mmap memory range of the user's choice.

doca_error_t doca_buf_inventory_buf_by_addr (doca_buf_inventory *inventory, doca_mmap *mmap, char *addr, size_t len, doca_buf **buf)

Allocate single element from buffer inventory and point it to the buffer defined by `addr` & `len` arguments.

Parameters

inventory
   The DOCA Buf inventory.
mmap
   DOCA memory map structure.
addr
   The start address of the payload.
len
   The length in bytes of the payload.
buf
   Doca buf allocated and initialized with args.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or if there is no suitable memory range for the given address and length.

▶ DOCA_ERROR_NOT_PERMITTED - if doca_mmap or doca_buf_inventory is un-started/ stopped.

▶ DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

doca_error_t doca_buf_inventory_buf_dup (doca_buf_inventory *inventory, const doca_buf *src_buf, doca_buf **dst_buf)

Duplicates content of the `buf` argument into element allocated from buffer inventory. (I.e., deep copy).

Parameters

inventory
   Buffer inventory structure that will hold the new doca_buf.
src_buf
   The DOCA buf to be duplicated.
dst_buf
   A duplicate DOCA Buf.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_NOT_PERMITTED - if src_buf mmap or input inventory unstarted/stopped or src_buf inventory extensions and the input inventory extensions are incompatible.
- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc new doca_buf from the given inventory.

doca_error_t doca_buf_inventory_create (const char *name, size_t num_elements, uint32_t extensions, doca_buf_inventory **buf_inventory)

Allocates buffer inventory with default/unset attributes.

Parameters

name
Name of created buffer inventory. The name of the buffer inventory must not exceed 31 characters.
num_elements
Initial number of elements in the inventory.
extensions
Bitmap of extensions enabled for the inventory described in doca_buf.h.
buf_inventory
Buffer inventory with default/unset attributes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_buf_inventory.
- ▶ DOCA_ERROR_UNKNOWN - failed to set doca_buf_inventory name.

Description

The returned object can be manipulated with doca_buf_inventory_property_set() API. Once all required attributes are set, it should be reconfigured and adjusted to meet the setting with doca_buf_inventory_start(). See doca_buf_inventory_start for the rest of the details.

doca_error_t doca_buf_inventory_destroy (doca_buf_inventory *inventory)

Destroy buffer inventory structure.

Parameters

inventory
 Buffer inventory structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_NOT_PERMITTED - if not all allocated elements had been returned to the inventory.

Description

Before calling this function all allocated elements should be returned back to the inventory.

doca_error_t doca_buf_inventory_property_get (doca_buf_inventory *inventory, doca_buf_inventory_property property, uint8_t *value, uint32_t size)

Get the value of a DOCA Inventory property.

Parameters

inventory
 The DOCA Buf inventory.
property
 The requested property to get. See enum doca_buf_inventory_property.
value
 The value of the property.
size
 The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_inventory_property_set (doca_buf_inventory *inventory, doca_buf_inventory_property property, const uint8_t *value, uint32_t size)

Set the value of a DOCA Inventory property.

Parameters

inventory
    The DOCA Buf inventory.

property
    The requested property to set. See enum doca_buf_inventory_property. Note: once an inventory object has been first started this functionality will not be availble.

value
    The new value of the property.

size
    The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_NOT_PERMITTED - if function is called after initial start of the inventory.

Description

> 📄 **Note:**
>
> Read only properties can't be set.

doca_error_t doca_buf_inventory_start (doca_buf_inventory *inventory)

Start element retrieval from inventory.

Parameters

inventory
    Buffer inventory structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Un-started/stopped buffer inventory rejects all attempts to retrieve element. For example doca_buf_inventory_buf_by_addr() will fail on stopped inventory regardless the number of free elements. Once buffer inventory object started it can provide following functionality:

▶ retrieval of free elements from the inventory. On first start verifies & finalizes the buffer inventory object configuration, the user can no longer set doca_buf_inventory properties.

doca_error_t doca_buf_inventory_stop (doca_buf_inventory *inventory)

Stop element retrieval from inventory.

Parameters

inventory
    Buffer inventory structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

No retrieval of elements with for stopped inventory. For example doca_buf_inventory_buf_by_addr() will fail on stopped inventory regardless the number of free elements.

2.3.3. Compatibility Management

Core

Lib to define compatibility with current version, define experimental Symbols.

To set a Symbol (or specifically a function) as experimental:

__DOCA_EXPERIMENTAL int func_declare(int param1, int param2);

To remove warnings of experimental compile with "-D DOCA_ALLOW_EXPERIMENTAL_API"

#define __DOCA_EXPERIMENTAL __attribute__((deprecated("Symbol is defined as experimental"), section(".text.experimental"))) DOCA_EXPORTED

To set a Symbol (or specifically a function) as experimental.

2.3.4.   DOCA Context

Core

DOCA CTX is the base class of every data-path library in DOCA. It is a specific library/SDK instance object providing abstract data processing functionality. The library exposes events and/or jobs that manipulate data.

struct doca_event

Activity completion event.

struct doca_job

Job structure describes request arguments for service provided by context.

doca_error_t doca_ctx_dev_add (doca_ctx *ctx, doca_dev *dev)

Add a device to a DOCA CTX.

Parameters

ctx
   The CTX to add the device to.
dev
   The device to add.

Returns

DOCA_SUCCESS - In case of success. Error code - On failure:

▶   DOCA_ERROR_INVALID_VALUE - received invalid input.

▶   DOCA_ERROR_BAD_STATE - CTX is started.

doca_error_t doca_ctx_dev_rm (doca_ctx *ctx, doca_dev *dev)

Remove a device from a context.

Parameters

ctx
  The CTX to remove the device from. Must already hold the device.

dev
  The device to remove.

Returns

DOCA_SUCCESS - In case of success. Error code - On failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_BAD_STATE - CTX is started.

doca_error_t doca_ctx_start (doca_ctx *ctx)

Finalizes all configurations, and starts the DOCA CTX.

Parameters

ctx
  The DOCA context to start.

Returns

DOCA_SUCCESS - In case of success. Error code - In case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

After starting the CTX, it can't be configured any further. Use doca_ctx_stop in order to reconfigure the CTX.

doca_error_t doca_ctx_stop (doca_ctx *ctx)

Stops the context allowing reconfiguration.

Parameters

ctx
  The DOCA context to stop.

Returns

DOCA_SUCCESS - In case of success. Error code - In case of failure:

▶   DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Once a context has started, it can't be configured any further. This method should be called in case the context needs to be configured after starting.

doca_error_t doca_ctx_workq_add (doca_ctx *ctx, doca_workq *workq)

Add a workQ to a context.

Parameters

ctx
   The library instance that will handle the jobs.
workq
   The WorkQ where you want to receive job completions.

Returns

DOCA_SUCCESS - In case of success. Error code - on failure:

▶   DOCA_ERROR_INVALID_VALUE - received invalid input.
▶   DOCA_ERROR_BAD_STATE - CTX is not started.
▶   DOCA_ERROR_UNKNOWN - received corrupted CTX.
▶   DOCA_ERROR_IN_USE - same WorkQ already added.

Description

This method adds a WorkQ to a context. Once a WorkQ has been added it will start accepting jobs defined by the CTX & retrieve events from the CTX. The jobs can be progressed using doca_workq_progress_retrieve().

doca_error_t doca_ctx_workq_rm (doca_ctx *ctx, doca_workq *workq)

Remove a DOCA WorkQ from a DOCA CTX.

Parameters

ctx
   The library instance containing the WorkQ.

workq
> The WorkQ to remove.

Returns

DOCA_SUCCESS - In case of success. Error code - on failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_UNKNOWN - received corrupted CTX.
- ▶ DOCA_ERROR_BAD_STATE - CTX is not started.
- ▶ DOCA_ERROR_NOT_PERMITTED - WorkQ does not exist within CTX.

Description

In order to remove the WorkQ from the CTX, the CTX must be stopped.

doca_error_t doca_workq_create (uint32_t depth, doca_workq **workq)

Creates empty DOCA WorkQ object with default attributes.

Parameters

depth
> The maximum number of inflight jobs.

workq
> The newly created WorkQ.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - invalid input received.
- ▶ DOCA_ERROR_NO_MEMORY - failed to allocate WorkQ.

Description

The returned WorkQ needs to be added to at least one DOCA CTX. Then the WorkQ can be used to progress jobs and to poll events exposed by the associated CTX.

doca_error_t doca_workq_destroy (doca_workq *workq)

Destroy a DOCA WorkQ.

Parameters

workq
> The WorkQ to destroy.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - invalid input received.

▶ DOCA_ERROR_IN_USE - WorkQ not removed from one of the doca_ctx.

Description

In order to destroy a WorkQ, at first needs to be removed from all DOCA CTXs using it.

doca_error_t doca_workq_progress_retrieve (doca_workq *workq, doca_event *ev, int flags)

Progress & retrieve single pending event.

Parameters

workq
    The WorkQ object to poll for events.

ev
    Event structure to be filled in case an event was received.

flags
    Flags for progress/retrival operations. A combination of enum doca_workq_retrieve_flags.

Returns

▶ DOCA_SUCCESS - on successful event retrieval. ev output argument is set.

▶ DOCA_ERROR_AGAIN - no event available (ev output argument not set), try again to make more progress.

▶ DOCA_ERROR_NOT_PERMITTED - the retrieved event is a failure event. The specific error is reported per action type.

▶ DOCA_ERROR_UNKNOWN - internal WorkQ error (ev output argument not set).

Description

Polling method for progress of submitted jobs and retrieval of events.

NOTE: for V1 retrieve supported for single event only.

doca_error_t doca_workq_property_get (const doca_workq *workq, doca_workq_property property, uint8_t *value, uint32_t size)

Get the value of a DOCA WorkQ property.

Parameters

workq
  The DOCA WorkQ.
property
  The requested property to get. See enum doca_workq_property.
value
  Where to write the current property value.
size
  The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_workq_property_set (doca_workq *workq, doca_workq_property property, const uint8_t *value, uint32_t size)

Set the value of a DOCA WorkQ property.

Parameters

workq
  The DOCA WorkQ.
property
  The requested property to set. See enum doca_workq_property.
value
  The new value of the property.
size
  The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_workq_submit (doca_workq *workq, doca_job *job)

Submit a job to a DOCA WorkQ.

Parameters

workq
  The DOCA WorkQ used for progress and retrieval of jobs.
job
  The job to submit, the job must be compatible with the WorkQ.

Returns

DOCA_SUCCESS - in case the job was submitted successfully, doca_workq_progress_retrieve() can be called next. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NO_MEMORY - in case the queue is full. See WorkQ depth.
- ▶ DOCA_ERROR_BAD_STATE - in case job->ctx is stopped.

Description

This method is used to submit a job to the WorkQ. The WorkQ should be added to the job->ctx via doca_ctx_workq_add() before job submission. Once a job has been submitted, it can be progressed using doca_workq_progress_retrieve() until the result is ready and retrieved.

#define DOCA_ACTION_SDK_RANGE 16

Power 2 single SDK/context action type range.

#define DOCA_MAX_NUM_CTX 1024

Maximum number of doca_ctx allowed within an application.

2.3.5.   DOCA Device

Core

DOCA Device library. For more details please refer to the user guide on DOCA devzone. The DOCA device represents an available processing unit backed by the HW or SW implementation.

enum doca_dev_remote_filter

Remote device filter by flavor

Multiple options possible but some are mutually exclusive.

Values

DOCA_DEV_REMOTE_FILTER_ALL = 0
DOCA_DEV_REMOTE_FILTER_NET = 1<<1

__DOCA_EXPERIMENTAL doca_devinfo *doca_dev_as_devinfo (doca_dev *dev)

Get local device info from device. This should be useful when wanting to query information about device after opening it, and destroying the devinfo list.

Parameters

dev

    The doca device instance.

Returns

The matching doca_devinfo instance in case of success, NULL in case dev is invalid.

doca_error_t doca_dev_close (doca_dev *dev)

Destroy allocated local device instance.

Parameters

dev

    The local doca device instance.

Returns

DOCA_SUCCESS - in case of success.

▶   DOCA_ERROR_IN_USE - failed to deallocate device resources.

doca_error_t doca_dev_open (doca_devinfo *devinfo, doca_dev **dev)

Initialize local device for use.

Parameters

devinfo

    The devinfo structure of the requested device.

dev

    Initialized local doca device instance on success. Valid on success only.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_UNKNOWN - received corrupted devinfo. Or missing RDMA driver.

Description

> 💬 **Note:**
>
> In case the same device was previously opened, then the same doca_dev instance is returned.

__DOCA_EXPERIMENTAL doca_devinfo_remote *doca_dev_remote_as_devinfo (doca_dev_remote *dev_remote)

Get remote device info from device. This should be useful when wanting to query information about device after opening it, and destroying the devinfo list.

Parameters

dev_remote
    The remote doca device instance.

Returns

The matching doca_devinfo_remote instance in case of success, NULL in case dev_remote is invalid.

doca_error_t doca_dev_remote_close (doca_dev_remote *dev)

Destroy allocated remote device instance.

Parameters

dev
    The remote doca device instance.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_IN_USE - failed to deallocate device resources.

doca_error_t doca_dev_remote_open (doca_devinfo_remote *devinfo, doca_dev_remote **dev_remote)

Initialize remote device for use.

Parameters

devinfo
   The devinfo structure of the requested device.
dev_remote
   Initialized remote doca device instance on success. Valid on success only.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_NO_MEMORY - failed to allocate memory for device.

doca_error_t doca_devinfo_list_create (doca_devinfodev_list, uint32_t *nb_devs)

Creates list of all available local devices.

Parameters

dev_list
   Pointer to array of pointers. Output can then be accessed as follows (*dev_list)[idx].
nb_devs
   Number of available local devices.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_UNKNOWN - failed to fetch device list. Maybe RDMA driver is missing?

▶ DOCA_ERROR_NO_MEMORY - failed to allocate enough space.

Description

Lists information about available devices, to start using the device you first have to call doca_dev_open(), while passing an element of this list. List elements become invalid once it has been destroyed.

💬 **Note:**

Returned list must be destroyed using doca_devinfo_list_destroy()

doca_error_t doca_devinfo_list_destroy (doca_devinfo **dev_list)

Destroy list of local device info structures.

Parameters

dev_list
   List to be destroyed.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_IN_USE - at least one device in the list is in a corrupted state.

Description

Destroys the list of device information, once the list has been destroyed, all elements become invalid.

doca_error_t doca_devinfo_property_get (const doca_devinfo *devinfo, doca_devinfo_property property, uint8_t *value, uint32_t size)

Get the value of a DOCA devinfo property.

Parameters

devinfo
   The device to query.
property
   The requested property to get. See enum doca_devinfo_property.
value
   Where to write the current property value.
size
   The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input, or size does not match the property size. See enum doca_devinfo_property.

▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this property.

▶ DOCA_ERROR_UNKNOWN - unknown error occured.

doca_error_t doca_devinfo_remote_list_create (doca_dev *dev, int filter, doca_devinfo_remotedev_list_remote, uint32_t *nb_devs_remote)

Create list of available remote devices accessible by dev.

Parameters

dev
Local device with access to representors.
filter
Bitmap filter of representor types. See enum doca_dev_remote_filter for more details.
dev_list_remote
Pointer to array of pointers. Output can then be accessed as follows (*dev_list_remote)[idx].
nb_devs_remote
Number of available remote devices.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_UNKNOWN - failed to fetch device list. Maybe RDMA driver is missing?

▶ DOCA_ERROR_NO_MEMORY - failed to allocate memory for list.

▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose remote devices.

Description

Returns all representors managed by the provided device. The provided device must be a local device. The representor may represent a network function attached to the host, or it can represent an emulated function attached to the host.

💬 **Note:**

Returned list must be destroyed using doca_devinfo_remote_list_destroy()

doca_error_t doca_devinfo_remote_list_destroy (doca_devinfo_remote **dev_list_remote)

Destroy list of remote device info structures.

Parameters

dev_list_remote
List to be destroyed.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_IN_USE - the doca_dev that created the list is in a corrupted state.

Description

Destroy list of remote device information, once the list has been destroyed, all elements of the list are considered invalid.

doca_error_t doca_devinfo_remote_property_get (const doca_devinfo_remote *devinfo_remote, doca_devinfo_remote_property property, uint8_t *value, uint32_t size)

Get the value of a DOCA remote devinfo property.

Parameters

devinfo_remote
The device to query.
property
The requested property to get. See enum doca_devinfo_remote_property.
value
Where to write the current property value.
size
The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input, or size does not match the property size. See enum doca_devinfo_remote_property.

▶ DOCA_ERROR_UNKNOWN - unknown error occured.

2.3.6. DOCA Error

Core

DOCA Error library. For more details please refer to the user guide on DOCA devzone.

const __DOCA_EXPERIMENTAL char *doca_get_error_name (doca_error_t error)

Returns the string representation of an error code name.

Parameters

error
  - Error code to convert to string.

Returns

char* pointer to a NULL-terminated string.

Description

Returns a string containing the name of an error code in the enum. If the error code is not recognized, "unrecognized error code" is returned.

const __DOCA_EXPERIMENTAL char *doca_get_error_string (doca_error_t error)

Returns the description string of an error code.

Parameters

error
  - Error code to convert to description string.

Returns

char* pointer to a NULL-terminated string.

Description

This function returns the description string of an error code. If the error code is not recognized, "unrecognized error code" is returned.

2.3.7. Logging Management

Core

Define functions for internal and external logging management

To add DOCA internal logging compile with "-D DOCA_LOGGING_ALLOW_DLOG"

enum DOCA_LOG_LEVEL

log levels

Values

DOCA_LOG_LEVEL_CRIT
    Critical log level
DOCA_LOG_LEVEL_ERROR
    Error log level
DOCA_LOG_LEVEL_WARNING
    Warning log level
DOCA_LOG_LEVEL_INFO
    Info log level
DOCA_LOG_LEVEL_DEBUG
    Debug log level

typedef (*log_flush_callback) (char* buffer)

logging backend flush() handler

__DOCA_EXPERIMENTAL void doca_log (uint32_t level, uint32_t source, int line, const char *format, ...)

Generates a log message.

Parameters

level
    Log level enum DOCA_LOG_LEVEL.
source
    The log source identifier defined by doca_log_source_register.
line
    The line number this log originated from.
format
    printf(3) arguments, format and variables.

Description

The log will be shown in the doca_log_stream_redirect (see default). This should not be used, please prefer using DOCA_LOG...

__DOCA_EXPERIMENTAL void doca_log_backend_level_set (doca_logger_backend *logger, uint32_t level)

Set the log level of a specific logger backend.

Parameters

logger
Logger backend to update.
level
Log level enum DOCA_LOG_LEVEL.

Description

Dynamically change the log level of the given logger backend, any log under this level will be shown.

__DOCA_EXPERIMENTAL doca_logger_backend *doca_log_create_buffer_backend (char *buffer, size_t capacity, log_flush_callback handler)

Create a logging backend with a char buffer stream.

Parameters

buffer
The char buffer (char *) for the logger's stream.
capacity
Maximal amount of chars that could be written to the stream.
handler
Handler to be called when the log record should be flushed from the stream.

Returns

struct doca_logger_backend * on success, NULL otherwise.

Description

Creates a new logging backend that will be added on top of the default logger. The logger will write each log record at the beginning of this buffer.

__DOCA_EXPERIMENTAL doca_logger_backend *doca_log_create_fd_backend (int fd)

Create a logging backend with an fd stream.

Parameters

fd

   The file descriptor (int) for the logger's backend.

Returns

struct doca_logger_backend * on success, NULL otherwise.

Description

Creates a new logging backend that will be added on top of the default logger.

__DOCA_EXPERIMENTAL doca_logger_backend *doca_log_create_file_backend (FILE *fptr)

Create a logging backend with a FILE* stream.

Parameters

fptr

   The FILE * for the logger's stream.

Returns

struct doca_logger_backend * on success, NULL otherwise.

Description

Creates a new logging backend that will be added on top of the default logger.

__DOCA_EXPERIMENTAL doca_logger_backend *doca_log_create_syslog_backend (const char *name)

Create a logging backend with a syslog output.

Parameters

name

   The syslog name for the logger's backend.

Returns

struct doca_logger_backend * on success, NULL otherwise.

Description

Creates a new logging backend that will be added on top of the default logger.

__DOCA_EXPERIMENTAL void __DOCA_EXPERIMENTAL void doca_log_developer (uint32_t level, uint32_t source, int line, const char *format, ...)

Generates a log message for DLOG operations.

Parameters

level
  Log level enum DOCA_LOG_LEVEL.
source
  The log source identifier defined by doca_log_source_register.
line
  The line number this log originated from.
format
  printf(3) arguments, format and variables.

Description

The log will be shown in the doca_log_stream_redirect (see default).

> 📝 **Note:**
>
> This function is thread safe.

__DOCA_EXPERIMENTAL uint16_t doca_log_get_bucket_time (void)

Get the timespan of the rate-limit bucket.

Returns

Time (in seconds) of the rate-limit bucket.

__DOCA_EXPERIMENTAL uint16_t doca_log_get_quantity (void)

Get the quantity of the rate-limit bucket.

Returns

Maximal number of log events for a rate-limit bucket.

__DOCA_EXPERIMENTAL uint32_t doca_log_global_level_get (void)

Get the log level of the default logger backend.

Returns

Log level enum DOCA_LOG_LEVEL.

Description

Dynamically query for the log level of the default logger backend, any log under this level will be shown.

__DOCA_EXPERIMENTAL void doca_log_global_level_set (uint32_t level)

Set the log level of the default logger backend.

Parameters

level
    Log level enum DOCA_LOG_LEVEL.

Description

Dynamically change the log level of the default logger backend, any log under this level will be shown.

__DOCA_EXPERIMENTAL int doca_log_rate_bucket_register (uint32_t source)

Register a new rate bucket.

Parameters

source
    The log source identifier defined by doca_log_source_register.

Returns

The bucket identifier. Negative for err.

Description

Will return the ID associated with the new bucket.

__DOCA_EXPERIMENTAL void __DOCA_EXPERIMENTAL void __DOCA_EXPERIMENTAL void doca_log_rate_limit (uint32_t level, uint32_t source, int line, uint32_t bucket_id, const char *format, ...)

Generates a log message with rate limit.

Parameters

level
    Log level enum DOCA_LOG_LEVEL.
source
    The log source identifier defined by doca_log_source_register.
line
    The line number this log originated from.
bucket_id
    The bucket identifier defined by doca_log_rate_bucket_register.
format
    printf(3) arguments, format and variables.

Description

The log will be shown in the doca_log_stream_redirect (see default). This should not be used, please prefer using DOCA_LOG_RATE_LIMIT...

__DOCA_EXPERIMENTAL void doca_log_set_bucket_time (const uint16_t bucket_time)

Set the timespan of the rate-limit bucket.

Parameters

bucket_time
    Time (in seconds) for the rate-limit bucket.

__DOCA_EXPERIMENTAL void doca_log_set_quantity (const uint16_t quantity)

Set the quantity of the rate-limit bucket.

Parameters

quantity

Maximal number of log events for a rate-limit bucket.

__DOCA_EXPERIMENTAL int doca_log_source_register (const char *source_name)

Register a log source.

Parameters

source_name

The string identifying the log source. Should be in an heirarchic form (i.e. DPI::Parser).

Returns

The log source identifier. Negative for err.

Description

Will return the ID associated with the log source. Log source name will be shown in the logs.

__DOCA_EXPERIMENTAL int doca_log_stream_redirect (FILE *stream)

Redirect the logger to a different stream.

Parameters

stream

Pointer to the stream.

Returns

0 on success, error code otherwise.

Description

Dynamically change the logger stream of the default logger backend. The default stream is stderr.

#define DOCA_DLOG do { \ } while (0)

Generates a development log message.

The DOCA_DLOG() is the main log function for development purposes logging. To show the logs, define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler. Consider using the specific level DOCA_LOG for better code readability (i.e. DOCA_DLOG_ERR).

#define DOCA_DLOG_CRIT DOCA_DLOG(CRIT, format)

Generates a CRITICAL development log message.

Will generate critical log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_DBG DOCA_DLOG(DEBUG, format)

Generates a DEBUG development log message.

Will generate debug log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_ERR DOCA_DLOG(ERROR, format)

Generates an ERROR development log message.

Will generate error log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_INFO DOCA_DLOG(INFO, format)

Generates an INFO development log message.

Will generate info log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_WARN DOCA_DLOG(WARNING, format)

Generates a WARNING development log message.

Will generate warning log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_LOG doca_log(DOCA_LOG_LEVEL_##level, log_id, __LINE__, format)

Generates a log message.

The DOCA_LOG() is the main log function for logging. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation. Consider using the specific level DOCA_LOG for better code readability (i.e. DOCA_LOG_ERR).

#define DOCA_LOG_CRIT DOCA_LOG(CRIT, format)

Generates a CRITICAL log message.

Will generate critical log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_DBG DOCA_LOG(DEBUG, format)

Generates a DEBUG log message.

Will generate debug log. This call affects the performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_ERR DOCA_LOG(ERROR, format)

Generates an ERROR log message.

Will generate error log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_INFO DOCA_LOG(INFO, format)

Generates an INFO log message.

Will generate info log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_RATE_LIMIT do { \ static int bucket_id = -1; \ if (bucket_id == -1) { \ bucket_id = doca_log_rate_bucket_register(log_id); \ } \ doca_log_rate_limit(DOCA_LOG_LEVEL_##level, log_id, __LINE__, bucket_id, format); \ } while (0)

Generates a log message with rate limit.

The DOCA_LOG_RATE_LIMIT calls DOCA_LOG with some rate limit. Implied to be used on hot paths.

#define DOCA_LOG_RATE_LIMIT_CRIT DOCA_LOG_RATE_LIMIT(CRIT, format)

Generates a CRITICAL rate limited log message.

#define DOCA_LOG_RATE_LIMIT_DBG DOCA_LOG_RATE_LIMIT(DEBUG, format)

Generates a DEBUG rate limited log message.

#define DOCA_LOG_RATE_LIMIT_ERR DOCA_LOG_RATE_LIMIT(ERROR, format)

Generates an ERROR rate limited log message.

#define DOCA_LOG_RATE_LIMIT_INFO DOCA_LOG_RATE_LIMIT(INFO, format)

Generates an INFO rate limited log message.

#define DOCA_LOG_RATE_LIMIT_WARN DOCA_LOG_RATE_LIMIT(WARNING, format)

Generates a WARNING rate limited log message.

#define DOCA_LOG_REGISTER static int log_id; \ /* Use the highest priority so other Ctors will be able to use the log */\ static void __attribute__((constructor(101), used)) __##__LINE__(void) \ { \ log_id = doca_log_source_register(#SOURCE); \ }

Registers log source on program start.

Should be used to register the log source. For example

DOCA_LOG_REGISTER(dpi)

void foo { DOCA_LOG_INFO("Message"); }

#define DOCA_LOG_WARN DOCA_LOG(WARNING, format)

Generates a WARNING log message.

Will generate warning log. This call affects the performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

2.3.8.   Memory Map

Core

The DOCA memory map provides a centralized repository and orchestration of several memory ranges registration for each device attached to the memory map.

typedef void (doca_mmap_memrange_free_cb_t)

Function to be called for each populated memory range on memory map destroy.

doca_error_t doca_mmap_create (const char *name, doca_mmap **mmap)

Allocates zero size memory map object with default/unset attributes.

Parameters

name
    Name of newly created doca_mmap. The name of the mmap must not exceed 31 characters.

mmap
    DOCA memory map structure with default/unset attributes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_mmap.

▶ DOCA_ERROR_UNKNOWN - failed to set doca_mmap name.

Description

The returned memory map object can be manipulated with doca_mmap_property_set() API.

Once all required mmap attributes set it should be reconfigured and adjusted to meet object size setting with doca_mmap_start() See doca_mmap_start for the rest of the details.

doca_error_t doca_mmap_create_from_export (const char *name, const uint8_t *export_desc, size_t export_desc_len, doca_dev *dev, doca_mmap **mmap)

Creates a memory map object representing memory ranges in remote system memory space.

Parameters

name
    Name of newly created DOCA memory map.
export_desc
    An export descriptor generated by doca_mmap_export.
export_desc_len
    Length in bytes of the export_desc.
dev
    A local device connected to the device that resides in the exported mmap.
mmap
    DOCA memory map granting access to remote memory.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or internal error. The following errors are internal and will occur if failed to produce new mmap from export descriptor:

▶ DOCA_ERROR_NO_MEMORY - if internal memory allocation failed.

▶ DOCA_ERROR_NOT_SUPPORTED

▶ DOCA_ERROR_NOT_PERMITTED

▶ DOCA_ERROR_UNKNOWN

Description

Once this function called on the object it considered as from_export.

> 📝 **Note:**
>
> : The created object not backed by local memory.

Limitation: Can only support mmap consisting of a single chunk.

doca_error_t doca_mmap_destroy (doca_mmap *mmap)

Destroy DOCA Memory Map structure.

Parameters

mmap
    The DOCA memory map structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
▶ DOCA_ERROR_NOT_PERMITTED - if there is a memory region pointed by one or more `struct doca_buf`, or if memory deregistration failed.

Description

Before calling this function all allocated buffers should be returned back to the mmap.

doca_error_t doca_mmap_dev_add (doca_mmap *mmap, doca_dev *dev)

Register DOCA memory map on a given device.

Parameters

mmap
    DOCA memory map structure.
dev
    DOCA Dev instance with appropriate capability.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
▶ DOCA_ERROR_NOT_PERMITTED - if doca_mmap status is invalid for this operation or memory registration failed.
▶ DOCA_ERROR_NO_MEMORY - if reached to DOCA_MMAP_MAX_NUM_DEVICES.

▶ DOCA_ERROR_IN_USE - if doca_dev already exists in doca_mmap.

Description

Should not be called on:

▶ un-started/stopped memory map object.

▶ exported/from_export memory map object.

doca_error_t doca_mmap_dev_rm (doca_mmap *mmap, doca_dev *dev)

Deregister given device from DOCA memory map.

Parameters

mmap
  DOCA memory map structure.
dev
  DOCA Dev instance that was previously added.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or doca_dev doesn't exists in doca_mmap.

▶ DOCA_ERROR_NOT_PERMITTED - if doca_mmap status is invalid for this operation or memory deregistration failed.

Description

Should not be called on:

▶ un-started/stopped memory map object.

▶ exported/from_export memory map object.

doca_error_t doca_mmap_export (doca_mmap *mmap, const doca_dev *dev, uint8_t **export_desc, size_t *export_desc_len)

Compose memory map representation for later import with doca_mmap_create_from_export() for one of the devices previously added to the memory map.

Parameters

mmap
DOCA memory map structure.

dev
Device previously added to the memory map via doca_mmap_dev_add().

export_desc
On successful return should have a pointer to the allocated blob containing serialized representation of the memory map object for the device provided as `dev`.

export_desc_len
Length in bytes of the export_desc.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or device does not exists in mmap.

- DOCA_ERROR_NOT_PERMITTED - if doca_mmap status is invalid for this operation. The following errors will occur if failed to produce export descriptor:

- DOCA_ERROR_NO_MEMORY - if failed to alloc memory for export_desc.

- DOCA_ERROR_NOT_SUPPORTED

- DOCA_ERROR_UNKNOWN

Description

Once this function called on the object it considered as exported.

Freeing memory buffer pointed by `*export_desc` is the caller responsibility.

Should not be called on:

- un-started/stopped memory map object.

- exported/from_export memory map object.

Limitation: Can only support mmap consisting of a single chunk.

doca_error_t doca_mmap_populate (doca_mmap *mmap, char *addr, size_t len, size_t pg_sz, doca_mmap_memrange_free_cb_t *free_cb, void *opaque)

Add memory range to DOCA memory map.

Parameters

mmap
DOCA memory map structure.

addr
Start address of the memory range to be populated.

len
The size of the memory range in bytes.

pg_sz
Page size alignment of the provided memory range.

free_cb
Callback function to free the populated memory range on memory map destroy.

opaque
Opaque value to be passed to free_cb once called.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶ DOCA_ERROR_NOT_PERMITTED - if doca_mmap status is invalid for this operation or device registration failed or addr and len intersect with an existing chunk.

▶ DOCA_ERROR_NO_MEMORY - if reached to DOCA_MMAP_MAX_NUM_CHUNKS, or memory allocation failed.

Description

Should not be called on:

▶ un-started/stopped memory map object.

▶ exported/from_export memory map object.

doca_error_t doca_mmap_property_get (doca_mmap *mmap, doca_mmap_property property, uint8_t *value, uint32_t size)

Get the value of a DOCA Memory Map property.

Parameters

mmap
   The DOCA memory map structure.
property
   The requested property to set. See enum doca_mmap_property.
value
   The current value of the property.
size
   The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_property_set (doca_mmap *mmap, doca_mmap_property property, const uint8_t *value, uint32_t size)

Set the value of a DOCA Memory Map property. Note: once a memory map object has been first started this functionality will not be available.

Parameters

mmap
   The DOCA memory map structure.
property
   The requested property to set. See enum doca_mmap_property.
value
   The new value of the property.
size
   The size of the property in bytes.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶   DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

▶   DOCA_ERROR_NOT_PERMITTED - if trying to set properties after first start of the mmap.

doca_error_t doca_mmap_start (doca_mmap *mmap)

Start DOCA Memory Map.

Parameters

mmap
    DOCA memory map structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- ▶ DOCA_ERROR_NO_MEMORY - if memory allocation failed.

Description

Un-started/stopped memory map object rejects all attempts to map doca_buf structures to its memory ranges. Once memory map object started it can provide following functionality:

- ▶ doca_mmap_dev_add() & doca_mmap_dev_rm()
- ▶ doca_mmap_populate() On first start verifies & finalizes the mmap object configuration, the user can no longer set mmap properties.

doca_error_t doca_mmap_stop (doca_mmap *mmap)

Stop DOCA Memory Map.

Parameters

mmap
    DOCA memory map structure.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Un-started/stopped memory map object rejects all attempts to map doca_buf structures to its memory ranges. For details see doca_mmap_start().

2.3.9.   Version Management

Core

Define functions to get the DOCA version, and compare against it.

const char *doca_version (void)

Function returning DOCA's (SDK) version string.

Returns

version string, using the format major.minor.patch.

Description

> 📝 **Note:**
>
> Represents the SDK version a project was compiled with.

const __DOCA_EXPERIMENTAL char *doca_version_runtime (void)

Function returning DOCA's (runtime) version string.

Returns

version string, using the format major.minor.patch.

Description

> 📝 **Note:**
>
> Represents the runtime version a project is linked against.

#define DOCA_CURRENT_VERSION_NUM DOCA_VERSION_NUM(DOCA_VER_MAJOR, DOCA_VER_MINOR, DOCA_VER_PATCH)

Macro of current version number for comparisons.

#define DOCA_VER_MAJOR 1

Major version number 0-255.

#define DOCA_VER_MINOR 3

Minor version number 0-255.

#define DOCA_VER_PATCH 12

Patch version number 0-9999.

#define DOCA_VER_STRING "1.3.0012"

DOCA Version String.

#define DOCA_VERSION_EQ_CURRENT (DOCA_VERSION_NUM(major, minor, patch) == DOCA_CURRENT_VERSION_NUM)

Check if the version specified is equal to current.

#define DOCA_VERSION_LTE_CURRENT (DOCA_VERSION_NUM(major, minor, patch) <= DOCA_CURRENT_VERSION_NUM)

Check if the version specified is less then or equal to current.

#define DOCA_VERSION_NUM ((size_t)((major) << 24 | (minor) << 16 | (patch)))

Macro of version number for comparisons.

2.4.   Comm Channel

DOCA Communication Channel library let you set a direct communication channel between the host and the DPU. The channel is run over RoCE/IB protocol and is not part of the TCP/IP stack. Please follow the programmer guide for usage instructions.

struct doca_comm_channel_init_attr

Configuration attributes for endpoint initialization.

typedef HANDLE doca_event_channel_t

endpoint notification file descriptor for blocking with epoll for recv ready event

< Windows

doca_error_t doca_comm_channel_ep_connect (doca_comm_channel_ep_t *local_ep, const char *name, doca_comm_channel_addr_t **peer_addr)

Client side Connect.

Parameters

local_ep
    handle for the endpoint created beforehand with doca_comm_channel_ep_create().

name
    identifies the service, as a SERVICE_NAME_MAX bytes size string.

peer_addr
    Output, handle to use for sending packets and recognize source of messages.

Returns

DOCA_SUCCESS on success. Errors: DOCA_ERROR_INVALID_VALUE if no ep object, name or peer_address pointer given. DOCA_ERROR_NOT_PERMITTED if the function was called on the service or the endpoint is already connected. DOCA_ERROR_NO_MEMORY if memory allocation failed. DOCA_ERROR_INITIALIZATION if initialization of ep connection failed. DOCA_ERROR_CONNECTION_ABORTED if connection failed for any reason (connections rejected or failed). DOCA_ERROR_UNKNOWN if an unknown error occured.

Description

This function available only for client side use. As part of the connection process, the client preforms a handshake with the service.

If the connect function is being called before the service perform listen with the same name the connection will fail.

doca_error_t doca_comm_channel_ep_create (doca_comm_channel_init_attr *attr, doca_comm_channel_ep_t **ep)

Create local endpoint The endpoint handle represents all the configuration needed for the channel to run. The user need to hold one endpoint for all actions with the CommChannel on his side.

Parameters

attr
    Attributes to use when initializing the endpoint resources and QPs.

ep
　　Output, handle to the newly created endpoint object.

Returns

If the creation is successfull, ep will point to the newly created endpoint and DOCA_SUCCESS will be returned. Errors: DOCA_ERROR_INVALID_VALUE if no ep pointer or no attribute object was given. DOCA_ERROR_NOT_PERMITTED if the given msgsize is bigger than supported max size. DOCA_ERROR_NO_MEMORY if memory allocation failed during ep creation. DOCA_ERROR_INITIALIZATION if initialization of ep failed. DOCA_ERROR_UNKNOWN if an unknown error occured.

doca_error_t doca_comm_channel_ep_destroy (doca_comm_channel_ep_t *ep)

Release endpoint handle.

Returns

DOCA_SUCCESS on success. Errors: DOCA_ERROR_NOT_CONNECTED if ep does not exist.

Description

Blocking until all queued messages are sent The function close the event_channel and release all internal resources. The doca_comm_channel_ep_disconnect() is included as part of the destroy process.

doca_error_t doca_comm_channel_ep_disconnect (doca_comm_channel_ep_t *local_ep, doca_comm_channel_addr_t *peer_addr)

Disconnect the endpoint from the remote peer. block until all resources related to peer address are freed new connection could be created on the endpoint.

Parameters

local_ep
　　handle for the endpoint created beforehand with doca_comm_channel_ep_create().
peer_addr
　　peer address to be disconnect from.

Returns

DOCA_SUCCESS on success. Errors: DOCA_ERROR_INVALID_VALUE if no ep was provided. DOCA_ERROR_NOT_CONNECTED if there is no connection.

doca_event_channel_t doca_comm_channel_ep_event_channel_get (doca_comm_channel_ep_t *local_ep)

Extract the event_channel handle for user's use When the user send/receive packets with non-blocking mode, this handle can be used to get interrupt when a new event happened, using epoll or similar function. The event channel is owned by the endpoint and release when calling doca_comm_channel_ep_destroy().

Parameters

local_ep
handle for the endpoint created beforehand with doca_comm_channel_ep_create().

doca_error_t doca_comm_channel_ep_listen (doca_comm_channel_ep_t *local_ep, const char *name)

Service side listen on all interafces.

Parameters

local_ep
handle for the endpoint created beforehand with doca_comm_channel_ep_create().
name
identifies the service, as a SERVICE_NAME_MAX bytes size string.

Returns

DOCA_SUCCESS on success Errors: DOCA_ERROR_INVALID_VALUE if no ep object or no name was given. DOCA_ERROR_NOT_PERMITTED if the function was called on the client side. DOCA_ERROR_NO_MEMORY if memory allocation failed. DOCA_ERROR_INITIALIZATION if initialization of service failed. DOCA_ERROR_CONNECTION_ABORTED if registration of service failed. DOCA_ERROR_UNKNOWN if an unknown error occured.

Description

The function opens new QP for each vhca_id it exposes to and wait for new connections. After calling this function the user should call doca_comm_channel_ep_recvfrom() in order to get new peers to communicate with.

This function available only for service side use.

doca_error_t doca_comm_channel_ep_recvfrom (doca_comm_channel_ep_t *local_ep, void *msg, size_t *len, int flags, doca_comm_channel_addr_t **peer_addr)

Receive message from connected client/service.

Parameters

local_ep
handle for the endpoint created beforehand with doca_comm_channel_ep_create().

msg
pointer to the buffer where the message should be stored.

len
input - maximum len of bytes in the msg buffer, output - len of actual received message.

flags
DOCA_CC_MSG_FLAG_DONTWAIT to return on any case or 0 to block when waiting on empty queue.

peer_addr
output, recieved message source address handle

Returns

DOCA_SUCCESS on successful receive. If a message was received, the value pointed by len will be updated with the number of bytes received. Errors: DOCA_ERROR_INVALID_VALUE if any of the parameters is NULL. DOCA_ERROR_NOT_CONNECTED if endpoint is service and listen was not called. DOCA_ERROR_AGAIN if the command or the endpoint is set to non-blocking mode and no message was received. DOCA_ERROR_UNKNOWN if an unknown error occured.

Description

On service side, doca_comm_channel_ep_recvfrom() also used for accepting new connection from clients.

doca_error_t doca_comm_channel_ep_sendto (doca_comm_channel_ep_t *local_ep, const void *msg, size_t len, int flags, doca_comm_channel_addr_t *peer_addr)

Send message to peer address. The connection to the wanted peer_address need to be established before sending the message.

Parameters

local_ep
    handle for the endpoint created beforehand with doca_comm_channel_ep_create().
msg
    pointer to the message to be sent.
len
    length in bytes of msg.
flags
    DOCA_CC_MSG_FLAG_DONTWAIT to return on any case or 0 to block when waiting for receiver to be ready.
peer_addr
    destination address handle of the send operation.

Returns

DOCA_SUCCESS on success. Errors: DOCA_ERROR_NOT_CONNECTED if no peer_address was supplied or no connection was found. DOCA_ERROR_INVALID_VALUE if the supplied len was larger than the msgsize given at ep creation or any of the input variables are null. DOCA_ERROR_AGAIN if the command or the endpoint is set to non-blocking mode and the receiver queue is full. DOCA_ERROR_UNKNOWN if an unknown error occured.

doca_error_t doca_comm_channel_peer_addr_user_data_get (doca_comm_channel_addr_t *peer_addr, uint64_t *user_data)

Extract 'user_context' from peer_addr handle. By default, the 'user_context' is set to 0 and can be change using doca_comm_channel_peer_addr_user_data_set().

Parameters

peer_addr
    Pointer to peer_addr to extract user_context from.

user_data
    Output param, will contain the extracted data.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_address or user_data is NULL.

doca_error_t doca_comm_channel_peer_addr_user_data_set (doca_comm_channel_addr_t *peer_addr, uint64_t user_context)

Save 'user_context' in peer_addr handle.

Parameters

peer_addr
    Pointer to peer_addr to set user_context to.
user_context
    Data to set for peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_address is NULL.

Description

Can be use by the user to identify the peer address got from doca_comm_channel_ep_recvfrom(). The user_context for new peers is initialized to 0.

#define SERVICE_NAME_MAX 120

Service name length includes the terminating null byte ('').

2.5. DOCA DMA engine

DOCA DMA library. For more details please refer to the user guide on DOCA devzone.

struct doca_dma_job_memcpy

struct doca_dma_memcpy_result

enum doca_dma_devinfo_caps

Possible DMA device capabilities.

Values

DOCA_DMA_CAP_NONE = 0
DOCA_DMA_CAP_HW_OFFLOAD = 1U<<0
    DMA HW offload is supported

enum doca_dma_job_types

Available jobs for DMA.

Values

DOCA_DMA_JOB_MEMCPY = DOCA_ACTION_DMA_FIRST+1

__DOCA_EXPERIMENTAL doca_ctx *doca_dma_as_ctx (doca_dma *dma)

Parameters

dma
    DMA instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert doca_dma instance into a generalised context for use with doca core objects.

doca_error_t doca_dma_create (doca_dma **dma)

Parameters

dma
    Pointer to pointer to be set to point to the created doca_dma instance.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - dma argument is a NULL pointer.

▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_dma.

▶ DOCA_ERROR_INITIALIZATION - failed to initialise a mutex.

Description

Create a DOCA DMA instance.

doca_error_t doca_dma_destroy (doca_dma *dma)

Parameters

dma
   Pointer to instance to be destroyed.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_dma.

▶ DOCA_ERROR_IN_USE - Unable to gain exclusive access to the dma instance.

▶ DOCA_ERROR_IN_USE - One or more work queues are still attached. These must be detached first.

doca_error_t doca_dma_devinfo_caps_get (doca_devinfo *devinfo, uint32_t *caps)

Parameters

devinfo
   The DOCA device information
caps
   DMA capabilities available through this device. see enum doca_dma_devinfo_caps.

Returns

DOCA_SUCCESS - in case at least one capability is supported. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

▶ DOCA_ERROR_UNKNOWN - failed to query device capability. Maybe old FW?

▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support any DMA capability.

Description

Check if given device is DMA capable.

2.6. Deep packet inspection

DOCA Deep packet inspection library. For more details please refer to the user guide on DOCA devzone.

struct doca_dpi_config_t

DPI init configuration.

struct doca_dpi_parsing_info

L2-L4 flow information.

struct doca_dpi_result

Dequeue result.

struct doca_dpi_sig_data

Extra signature data.

struct doca_dpi_sig_info

Signature info.

struct doca_dpi_stat_info

DPI statistics.

enum doca_dpi_dequeue_status_t

Status of dequeue operation.

Values

DOCA_DPI_DEQ_NA
    No DPI enqueued jobs done, or no packets to dequeue
DOCA_DPI_DEQ_READY
    DPI Job and result is valid
DOCA_DPI_DEQ_NA
    No DPI enqueued jobs done, or no packets to dequeue
DOCA_DPI_DEQ_READY

DPI Job and result is valid

enum doca_dpi_enqueue_status_t

Status of enqueue operation.

Values

DOCA_DPI_ENQ_PROCESSING
Packet enqueued for processing
DOCA_DPI_ENQ_PACKET_EMPTY
No payload, packet was not queued
DOCA_DPI_ENQ_BUSY
Packet cannot be enqueued, queue is full
DOCA_DPI_ENQ_INVALID_DB
load_signatures failed, or was never called
DOCA_DPI_ENQ_INTERNAL_ERR
Other system errors possible
DOCA_DPI_ENQ_PROCESSING
Packet enqueued for processing
DOCA_DPI_ENQ_PACKET_EMPTY
No payload, packet was not queued
DOCA_DPI_ENQ_BUSY
Packet cannot be enqueued, queue is full
DOCA_DPI_ENQ_INVALID_DB
load_signatures failed, or was never called
DOCA_DPI_ENQ_INTERNAL_ERR
Other system errors possible

enum doca_dpi_flow_status_t

Status of enqueued entry.

Values

DOCA_DPI_STATUS_LAST_PACKET = 1<<1
Indicates there are no more packets in queue from this flow.
DOCA_DPI_STATUS_DESTROYED = 1<<2
Indicates flow was destroyed while being processed
DOCA_DPI_STATUS_NEW_MATCH = 1<<3
Indicates flow was matched on current dequeue
DOCA_DPI_STATUS_LAST_PACKET = 1<<1
Indicates there are no more packets in queue from this flow.
DOCA_DPI_STATUS_DESTROYED = 1<<2
Indicates flow was destroyed while being processed
DOCA_DPI_STATUS_NEW_MATCH = 1<<3

Indicates flow was matched on current dequeue

enum doca_dpi_sig_action_t

Signature action. Some signatures may come with an action.

Values

DOCA_DPI_SIG_ACTION_NA
Action not available for signature
DOCA_DPI_SIG_ACTION_ALERT
Alert
DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed
DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped
DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet
DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation
DOCA_DPI_SIG_ACTION_NA
Action not available for signature
DOCA_DPI_SIG_ACTION_ALERT
Alert
DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed
DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped
DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet
DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation

__DOCA_EXPERIMENTAL int doca_dpi_dequeue (doca_dpi_ctx *ctx, uint16_t dpi_q, doca_dpi_result *result)

Dequeues packets after processing.

Parameters

ctx
   The DPI context.
dpi_q
   The DPI queue from which to dequeue the flows' packets.
result
   Output, matching result.

Returns

doca_dpi_dequeue_status_t if successful, error code otherwise

Description

Only packets enqueued for processing will be returned by this API. Packets will return in the order they were enqueued.

__DOCA_EXPERIMENTAL void doca_dpi_destroy (doca_dpi_ctx *ctx)

Free the DPI memory and releases the regex engine.

Parameters

ctx
   DPI context to destroy.

__DOCA_EXPERIMENTAL int doca_dpi_enqueue (doca_dpi_flow_ctx *flow_ctx, rte_mbuf *pkt, bool initiator, uint32_t payload_offset, void *user_data)

Enqueue a new DPI job for processing.

Parameters

flow_ctx
   The flow context handler.

pkt

    The mbuf to be processed.

initiator

    Indicates to which direction the packet belongs. 1 - if the packet arrives from client to server. 0 - if the packet arrives from server to client. Typically, the first packet will arrive from the initiator (client).

payload_offset

    Indicates where the packet's payload begins.

user_data

    Private user data to b returned when the DPI job is dequeued.

Returns

doca_dpi_enqueue_status_t or other error code.

Description

This function is thread-safe per queue. For best performance it should always be called from the same thread/queue on which the flow was created. See Multithreading section of the DPI Programming Guide for more details.

Once a packet is enqueued, user must not change, reuse or free the mbuf while it is being processed. See "Packet Ownership" section of the DPI Programming Guide for more details.

The injected packet has to be stripped of FCS. A packet will not be enqueued if:

▶   Payload length = 0

__DOCA_EXPERIMENTAL doca_dpi_flow_ctx *doca_dpi_flow_create (doca_dpi_ctx *ctx, uint16_t dpi_q, const doca_dpi_parsing_info *parsing_info, int *error, doca_dpi_result *result)

Creates a new flow on a queue.

Parameters

ctx

    The DPI context.

dpi_q

    The DPI queue on which to create the flows

parsing_info

    L3/L4 information.

error

    Output, Negative if error occurred.

result

    Output, If flow was matched based on the parsing info, result->matched will be true.

Returns

NULL on error.

Description

Must be called before enqueuing any new packet. A flow must not be created on 2 different queues.

__DOCA_EXPERIMENTAL void doca_dpi_flow_destroy (doca_dpi_flow_ctx *flow_ctx)

Destroys a flow on a queue.

Parameters

flow_ctx

    The flow context to destroy.

Description

Should be called when a flow is terminated or times out

__DOCA_EXPERIMENTAL int doca_dpi_flow_match_get (const doca_dpi_flow_ctx *flow_ctx, doca_dpi_result *result)

Query a flow's match.

Parameters

flow_ctx

    The flow context of the flow to be queried.

result

    Output, latest match on this flow. Only "matched" and "info" fields in the result parameter are valid.

Returns

0 on success, error code otherwise.

__DOCA_EXPERIMENTAL doca_dpi_ctx *doca_dpi_init (const doca_dpi_config_t *config, int *error)

Initialize the DPI library.

Parameters

config

    See doca_dpi_config_t for details.

error

    Output error, negative value indicates an error.

Returns

doca_dpi_ctx - dpi opaque context, NULL on error.

Description

This function must be invoked first before any function in the API. It should be invoked once per process. This call will probe the first regex device it finds (0).

__DOCA_EXPERIMENTAL int doca_dpi_load_signatures (doca_dpi_ctx *ctx, const char *cdo_file)

Loads the cdo file.

Parameters

ctx

    The DPI context.

cdo_file

    CDO file created by the DPI compiler.

Returns

0 on success, error code otherwise.

Description

The cdo file contains signature information. The cdo file must be loaded before any enqueue call.

Database update: When a new signatures database is available, the user may call this function again. The newly loaded CDO must contain the signatures of the previously loaded CDO or result will be undefined.

__DOCA_EXPERIMENTAL int doca_dpi_signature_get (const doca_dpi_ctx *ctx, uint32_t sig_id, doca_dpi_sig_data *sig_data)

Returns a specific sig info.

Parameters

ctx
    The DPI context.
sig_id
    The signature ID.
sig_data
    Output of the sig metadata.

Returns

0 on success, error code otherwise.

__DOCA_EXPERIMENTAL int doca_dpi_signatures_get (const doca_dpi_ctx *ctx, doca_dpi_sig_data **sig_data)

Returns all signatures.

Parameters

ctx
    The DPI context.
sig_data
    Output of the sig data.

Returns

Number of signatures on success, error code otherwise.

Description

It is the responsibility of the user to free the array. Because this function copies all the sig info, it is highly recommended to call this function only once after loading the database, and not during packet processing.

__DOCA_EXPERIMENTAL void doca_dpi_stat_get (const doca_dpi_ctx *ctx, bool clear, doca_dpi_stat_info *stats)

Returns DPI statistics.

Parameters

ctx
　　The DPI context.
clear
　　Clear the statistics after fetching them.
stats
　　Output struct containing the statistics.

2.7.　　Remote deep packet inspection (grpc)

DOCA gRPC API for on-host clients to remote use of deep packet inspection library. For more details please refer to the user guide on DOCA devzone.

struct doca_dpi_grpc_generic_packet

Generic packet that holds payload or a whole packet as segment.

struct doca_dpi_grpc_result

Dequeue result.

struct doca_dpi_parsing_info

L2-L4 flow information.

struct doca_dpi_sig_data

Extra signature data.

struct doca_dpi_sig_info

Signature info.

struct doca_dpi_stat_info

DPI statistics.

enum doca_dpi_dequeue_status_t

Status of dequeue operation.

Values

DOCA_DPI_DEQ_NA
    No DPI enqueued jobs done, or no packets to dequeue
DOCA_DPI_DEQ_READY
    DPI Job and result is valid
DOCA_DPI_DEQ_NA
    No DPI enqueued jobs done, or no packets to dequeue
DOCA_DPI_DEQ_READY
    DPI Job and result is valid

enum doca_dpi_enqueue_status_t

Status of enqueue operation.

Values

DOCA_DPI_ENQ_PROCESSING
    Packet enqueued for processing
DOCA_DPI_ENQ_PACKET_EMPTY
    No payload, packet was not queued

DOCA_DPI_ENQ_BUSY
    Packet cannot be enqueued, queue is full
DOCA_DPI_ENQ_INVALID_DB
    load_signatures failed, or was never called
DOCA_DPI_ENQ_INTERNAL_ERR
    Other system errors possible
DOCA_DPI_ENQ_PROCESSING
    Packet enqueued for processing
DOCA_DPI_ENQ_PACKET_EMPTY
    No payload, packet was not queued
DOCA_DPI_ENQ_BUSY
    Packet cannot be enqueued, queue is full
DOCA_DPI_ENQ_INVALID_DB
    load_signatures failed, or was never called
DOCA_DPI_ENQ_INTERNAL_ERR
    Other system errors possible

enum doca_dpi_flow_status_t

Status of enqueued entry.

Values

DOCA_DPI_STATUS_LAST_PACKET = 1<<1
    Indicates there are no more packets in queue from this flow.
DOCA_DPI_STATUS_DESTROYED = 1<<2
    Indicates flow was destroyed while being processed
DOCA_DPI_STATUS_NEW_MATCH = 1<<3
    Indicates flow was matched on current dequeue
DOCA_DPI_STATUS_LAST_PACKET = 1<<1
    Indicates there are no more packets in queue from this flow.
DOCA_DPI_STATUS_DESTROYED = 1<<2
    Indicates flow was destroyed while being processed
DOCA_DPI_STATUS_NEW_MATCH = 1<<3
    Indicates flow was matched on current dequeue

enum doca_dpi_sig_action_t

Signature action. Some signatures may come with an action.

Values

DOCA_DPI_SIG_ACTION_NA
    Action not available for signature
DOCA_DPI_SIG_ACTION_ALERT
    Alert

DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed
DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped
DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet
DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation
DOCA_DPI_SIG_ACTION_NA
Action not available for signature
DOCA_DPI_SIG_ACTION_ALERT
Alert
DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed
DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped
DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet
DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet
DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation

__DOCA_EXPERIMENTAL int doca_dpi_grpc_dequeue (doca_dpi_ctx *ctx, uint16_t dpi_queue, doca_dpi_grpc_result *result)

Dequeues packets after processing.

Parameters

ctx
The DPI context.
dpi_queue
The DPI queue from which to dequeue the flows' packets.
result
Output, matching result.

Returns

doca_dpi_dequeue_status_t if successful, error code otherwise.

Description

Only packets enqueued for processing will be returned by this API. Packets will return in the order they were enqueued.

__DOCA_EXPERIMENTAL void doca_dpi_grpc_destroy (doca_dpi_ctx *ctx)

Close the connection to the DPI service and free the connection memory. This call doesn't free created flow contexts, make sure to destroy them beforehand.

Parameters

ctx

    The DPI context.

__DOCA_EXPERIMENTAL int doca_dpi_grpc_enqueue (doca_dpi_flow_ctx *flow_ctx, doca_dpi_grpc_generic_packet *pkt, bool initiator, uint32_t payload_offset, void *user_data, size_t user_data_len)

Enqueue a new DPI job for processing.

Parameters

flow_ctx

    The flow context handler.

pkt

    The packet as binary buffer to be processed.

initiator

    Indicates to which direction the packet belongs. 1 - if the packet arrives from client to server. 0 - if the packet arrives from server to client. Typically, the first packet will arrive from the initiator (client).

payload_offset

    Indicates where the packet's payload begins.

user_data

    Private user data to be returned when the DPI job is dequeued.

user_data_len
    The length of the user_data param.

Returns

doca_dpi_enqueue_status_t or other negative error code.

Description

This function is thread-safe per queue. See Multithreading section of the DPI Programming Guide for more details.

See "Packet Ownership" section of the DPI Programming Guide for more details.

The injected packet has to be stripped of FCS. A packet will not be enqueued if:

▶ Payload length = 0

__DOCA_EXPERIMENTAL doca_dpi_flow_ctx *doca_dpi_grpc_flow_create (doca_dpi_ctx *ctx, uint16_t dpi_q, const doca_dpi_parsing_info *parsing_info, int *error, doca_dpi_grpc_result *result)

Creates a new flow on a queue.

Parameters

ctx
    The DPI context.
dpi_q
    The DPI queue on which to create the flows
parsing_info
    L3/L4 information.
error
    Output, Negative if error occurred.
result
    Output, If flow was matched based on the parsing info, result->matched will be true.

Returns

NULL on error.

Description

Must be called before enqueuing any new packet. A flow must not be created on 2 different queues.

__DOCA_EXPERIMENTAL void doca_dpi_grpc_flow_destroy (doca_dpi_flow_ctx *ctx)

Destroys a flow on a queue.

Parameters

ctx

    The flow context to destroy.

Description

Should be called when a flow is terminated or times out

__DOCA_EXPERIMENTAL int doca_dpi_grpc_flow_match_get (const doca_dpi_flow_ctx *flow_ctx, doca_dpi_grpc_result *result)

Query a flow's match.

Parameters

flow_ctx

    The flow context of the flow to be queried.

result

    Output, latest match on this flow. Only "matched" and "info" fields in the result parameter are valid.

Returns

0 on success, error code otherwise.

__DOCA_EXPERIMENTAL doca_dpi_ctx *doca_dpi_grpc_init (const char *server_address, int *error)

Initialize a connection to the DPI gRPC service.

Parameters

server_address
    String representing the service ip, i.e. "127.0.0.1" or "192.168.100.3:5050". If no port is provided, it will use the service default port.
error
    Output error, negative value indicates an error.

Returns

doca_dpi_ctx - dpi opaque context, NULL on error.

Description

This function must be invoked first before any function in the API. It should be invoked once per process.

__DOCA_EXPERIMENTAL int doca_dpi_grpc_load_signatures (doca_dpi_ctx *ctx, const char *path_to_cdo)

Loads the cdo file.

Parameters

ctx
    The DPI context.
path_to_cdo
    Path on the DPU to the CDO file created by the DPI compiler.

Returns

0 on success, error code otherwise.

Description

The cdo file contains signature information. The cdo file must be loaded before any enqueue call.

Database update: When a new signatures database is available, the user may call this function again. The newly loaded CDO must contain the signatures of the previously loaded CDO or result will be undefined.

__DOCA_EXPERIMENTAL int doca_dpi_grpc_signature_get (const doca_dpi_ctx *ctx, uint32_t sig_id, doca_dpi_sig_data *sig_data)

Returns a specific sig info.

Parameters

ctx
    The DPI context.
sig_id
    The signature ID.
sig_data
    Output of the sig metadata.

Returns

0 on success, error code otherwise.

__DOCA_EXPERIMENTAL int doca_dpi_grpc_signatures_get (const doca_dpi_ctx *ctx, doca_dpi_sig_data **sig_data)

Returns all signatures.

Parameters

ctx
    The DPI context.
sig_data
    Output of the sig data.

Returns

Number of signatures on success, error code otherwise.

Description

It is the responsibility of the user to free the array. Because this function copies all the sig info, it is highly recommended to call this function only once after loading the database, and not during packet processing.

__DOCA_EXPERIMENTAL void doca_dpi_grpc_stat_get (const doca_dpi_ctx *ctx, bool clear, doca_dpi_stat_info *stats)

Returns DPI statistics.

Parameters

ctx
> The DPI context.

clear
> Clear the statistics after fetching them.

stats
> Output struct containing the statistics.

#define GENERAL_ERRORCODE -1

Unclassified error code for a general error which information is printed to the log.

#define IPV6_ADDER_LEN 16

Length of IPv6 address.

2.8.   Flow

DOCA HW offload flow library. For more details please refer to the user guide on DOCA devzone.

struct doca_flow_action_desc

action description

struct doca_flow_action_descs

action descriptions

struct doca_flow_action_descs_packet

packet action descriptions

struct doca_flow_action_field

extended modification action

struct doca_flow_actions

doca flow actions information

struct doca_flow_aged_query

aged flow query callback context

struct doca_flow_cfg

doca flow global configuration

struct doca_flow_encap_action

doca flow encap data information

struct doca_flow_error

doca flow error message struct

struct doca_flow_fwd

forwarding configuration

struct doca_flow_match

doca flow matcher information

struct doca_flow_monitor

doca monitor action configuration

struct doca_flow_pipe_cfg

pipeline configuration

struct doca_flow_port_cfg

doca flow port configuration

struct doca_flow_query

flow query result

struct doca_flow_resource_meter_cfg

doca flow meter resource configuration

struct doca_flow_resources

doca flow resource quota

struct doca_flow_shared_resource_cfg

doca flow shared resource configuration

enum doca_flow_action_type

action type enumeration

Values

DOCA_FLOW_ACTION_AUTO = 0
DOCA_FLOW_ACTION_CONSTANT
DOCA_FLOW_ACTION_SET
DOCA_FLOW_ACTION_ADD
DOCA_FLOW_ACTION_COPY
DOCA_FLOW_ACTION_MAX

enum doca_flow_entry_op

doca flow entry operation

Values

DOCA_FLOW_ENTRY_OP_ADD
    Add entry
DOCA_FLOW_ENTRY_OP_DEL
    Delete entry

enum doca_flow_entry_status

doca flow entry status

Values

DOCA_FLOW_ENTRY_STATUS_IN_PROCESS
DOCA_FLOW_ENTRY_STATUS_SUCCESS
DOCA_FLOW_ENTRY_STATUS_ERROR

enum doca_flow_error_type

doca flow error type define

Values

DOCA_ERROR_UNKNOWN
   Unknown error
DOCA_ERROR_UNSUPPORTED
   Operation unsupported
DOCA_ERROR_INVALID_PARAM
   Invalid parameter
DOCA_ERROR_PIPE_BUILD_ITEM
   Build pipe match items error
DOCA_ERROR_PIPE_MODIFY_ITEM
   Modify pipe match items error
DOCA_ERROR_PIPE_BUILD_ACTION
   Build pipe actions error
DOCA_ERROR_PIPE_MODIFY_ACTION
   Modify pipe actions error
DOCA_ERROR_PIPE_BUILD_FWD
   Build pipe fwd error
DOCA_ERROR_FLOW_CREATE
   Flow creation error
DOCA_ERROR_FLOW_DESTROY
   Flow destroy error
DOCA_ERROR_OOM
   Out of memory
DOCA_ERROR_PORT
   Port error
DOCA_ERROR_VERIFY_CONFIG
   Verification error

enum doca_flow_flags_type

doca flow flags type

Values

DOCA_FLOW_NO_WAIT = 0
  entry will not be buffered
DOCA_FLOW_WAIT_FOR_BATCH = (1<<0)
  entry will be buffered

enum doca_flow_fwd_type

forwarding action type

Values

DOCA_FLOW_FWD_NONE = 0
  No forward action be set
DOCA_FLOW_FWD_RSS
  Forwards packets to rss
DOCA_FLOW_FWD_PORT
  Forwards packets to one port
DOCA_FLOW_FWD_PIPE
  Forwards packets to another pipe
DOCA_FLOW_FWD_DROP
  Drops packets

enum doca_flow_match_tcp_flags

doca flow match flags

Values

DOCA_FLOW_MATCH_TCP_FLAG_FIN = (1<<0)
  match tcp packet with Fin flag
DOCA_FLOW_MATCH_TCP_FLAG_SYN = (1<<1)
  match tcp packet with Syn flag
DOCA_FLOW_MATCH_TCP_FLAG_RST = (1<<2)
  match tcp packet with Rst flag
DOCA_FLOW_MATCH_TCP_FLAG_PSH = (1<<3)
  match tcp packet with Psh flag
DOCA_FLOW_MATCH_TCP_FLAG_ACK = (1<<4)
  match tcp packet with Ack flag
DOCA_FLOW_MATCH_TCP_FLAG_URG = (1<<5)
  match tcp packet with Urg flag
DOCA_FLOW_MATCH_TCP_FLAG_ECE = (1<<6)
  match tcp packet with Urg flag
DOCA_FLOW_MATCH_TCP_FLAG_CWR = (1<<7)
  match tcp packet with Urg flag

enum doca_flow_pipe_type

doca flow pipe type

Values

DOCA_FLOW_PIPE_BASIC
  Flow pipe
DOCA_FLOW_PIPE_CONTROL
  Control pipe

enum doca_flow_port_type

doca flow port type

Values

DOCA_FLOW_PORT_DPDK_BY_ID
  dpdk port by mapping id

enum doca_flow_shared_resource_type

Shared resource supported types.

Values

DOCA_FLOW_SHARED_RESOURCE_METER
  Shared meter type
DOCA_FLOW_SHARED_RESOURCE_COUNT
  Shared counter type
DOCA_FLOW_SHARED_RESOURCE_MAX
  Shared max supported types

enum doca_rss_type

rss offload types

Values

DOCA_FLOW_RSS_IP = (1<<0)
  rss by ip head
DOCA_FLOW_RSS_UDP = (1<<1)
  rss by udp head
DOCA_FLOW_RSS_TCP = (1<<2)
  rss by tcp head

typedef (*doca_flow_entry_process_cb) (doca_flow_pipe_entry* entry, enum doca_flow_entry_status status, enum doca_flow_entry_op op, void* user_ctx)

doca flow entry process callback

struct __attribute__ ((__packed__))

doca flow meta data

Description

Meta data known as scratch data can be used to match or modify within pipes. Meta data can be set with value in previous pipes and match in later pipes. User can customize meta data structure as long as overall size doens't exceed limit. To match meta data, mask must be specified when creating pipe. Struct must be aligned to 32 bits. No initial value for Meta data, must match after setting value.

__DOCA_EXPERIMENTAL doca_flow_pipe_entry *doca_flow_control_pipe_add_entry (uint16_t pipe_queue, uint8_t priority, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_fwd *fwd, doca_flow_error *error)

Add one new entry to a control pipe.

Parameters

pipe_queue
    Queue identifier.
priority
    Priority value.
pipe
    Pointer to pipe.
match
    Pointer to match, indicate specific packet match information.
match_mask
    Pointer to match mask information.

fwd
> Pointer to fwd actions.

error
> Output error, set doca_flow_error for details.

Returns

Pipe entry handler on success, NULL otherwise and error is set.

Description

Refer to doca_flow_pipe_add_entry.

__DOCA_EXPERIMENTAL doca_flow_pipe *doca_flow_create_pipe (const doca_flow_pipe_cfg *cfg, const doca_flow_fwd *fwd, const doca_flow_fwd *fwd_miss, doca_flow_error *error)

Create one new pipe.

Parameters

cfg
> Pipe configuration.

fwd
> Fwd configuration for the pipe.

fwd_miss
> Fwd_miss configuration for the pipe. NULL for no fwd_miss. When creating a pipe if there is a miss and fwd_miss configured, packet steering should jump to it.

error
> Output error, set doca_flow_error for details.

Returns

Pipe handler on success, NULL otherwise and error is set.

Description

Create new pipeline to match and offload specific packets, the pipe configuration includes the following components:

match: Match one packet by inner or outer fields. match_mask: The mask for the matched items. actions: Includes the modify specific packets fields, Encap and Decap actions. monitor: Includes Count, Age, and Meter actions. fwd: The destination of the matched action, include RSS, Hairpin, Port, and Drop actions.

This API will create the pipe, but would not start the HW offload.

__DOCA_EXPERIMENTAL void doca_flow_destroy (void)

Destroy the doca flow.

Description

Release all the resources used by doca flow.

Must be invoked at the end of the application, before it exits.

__DOCA_EXPERIMENTAL void doca_flow_destroy_pipe (uint16_t port_id, doca_flow_pipe *pipe)

Destroy one pipe.

Parameters

port_id
　　Port_id of the port.
pipe
　　Pointer to pipe.

Description

Destroy the pipe, and the pipe entries that match this pipe.

__DOCA_EXPERIMENTAL void doca_flow_destroy_port (uint16_t port_id)

Destroy a doca port.

Parameters

port_id
　　Port id of the port.

Description

Destroy the doca port, free all resources of the port.

__DOCA_EXPERIMENTAL int doca_flow_entries_process (doca_flow_port *port, uint16_t pipe_queue, uint64_t timeout, uint32_t max_processed_entries)

Process entries in queue.

Parameters

port
   Port
pipe_queue
   Queue identifier.
timeout
   Max time in micro seconds for this function to process entries. Process once if timeout is 0
max_processed_entries
   Flow entries number to process If it is 0, it will proceed until timeout.

Returns

> 0: the number of entries processed 0: no entries are processed negative value: failure

Description

The application must invoke this function in order to complete the flow rule offloading and to receive the flow rule operation status.

doca_flow_entry_status doca_flow_entry_get_status (doca_flow_pipe_entry *entry)

Get entry's status.

Parameters

entry
   pipe entry

Returns

entry's status

__DOCA_EXPERIMENTAL int doca_flow_handle_aging (doca_flow_port *port, uint16_t queue, uint64_t quota, doca_flow_aged_query *entries, int len)

Handle aging of flows in queue.

Parameters

port
Port to handle aging
queue
Queue identifier.
quota
Max time quota in micro seconds for this function to handle aging.
entries
User input entries array for the aged flows.
len
User input length of entries array.

Returns

> 0 the number of aged flows filled in entries array. 0 no aged entries in current call. -1 full cycle done.

Description

Go over all flows and release aged flows from being tracked. The entries array will be filled with aged flows.

Since the number of flows can be very large, it can take a significant amount of time to go over all flows so this function is limited by time quota, which means it might return without handling all flows which requires the user to call it again. Once a full cycle is done this function will return -1.

__DOCA_EXPERIMENTAL int doca_flow_init (const doca_flow_cfg *cfg, doca_flow_error *error)

Initialize the doca flow.

Parameters

cfg
Port configuration, see doca_flow_cfg for details.
error
Output error, set doca_flow_error for details.

Returns

0 on success, a negative errno value otherwise and error is set.

Description

This is the global initialization function for doca flow. It initializes all resources used by doca flow.

Must be invoked first before any other function in this API. this is a one time call, used for doca flow initialization and global configurations.

__DOCA_EXPERIMENTAL doca_flow_pipe_entry *doca_flow_pipe_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, uint32_t flags, void *usr_ctx, doca_flow_error *error)

Add one new entry to a pipe.

Parameters

pipe_queue
    Queue identifier.
pipe
    Pointer to pipe.
match
    Pointer to match, indicate specific packet match information.
actions
    Pointer to modify actions, indicate specific modify information.
monitor
    Pointer to monitor actions.
fwd
    Pointer to fwd actions.
flags
    Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.
usr_ctx
    Pointer to user context.
error
    Output error, set doca_flow_error for details.

Returns

Pipe entry handler on success, NULL otherwise and error is set.

Description

When a packet matches a single pipe, will start HW offload. The pipe only defines which fields to match. When offloading, we need detailed information from packets, or we need to set some specific actions that the pipe did not define. The parameters include:

match: The packet detail fields according to the pipe definition. actions: The real actions according to the pipe definition. monitor: Defines the monitor actions if the pipe did not define it. fwd: Define the forward action if the pipe did not define it.

This API will do the actual HW offload, with the information from the fields of the input packets.

__DOCA_EXPERIMENTAL int doca_flow_pipe_rm_entry (uint16_t pipe_queue, void *usr_ctx, doca_flow_pipe_entry *entry)

Free one pipe entry.

Parameters

pipe_queue
    Queue identifier.
usr_ctx
    The pointer to user context.
entry
    The pipe entry to be removed.

Returns

0 on success, negative on failure.

Description

This API will free the pipe entry and cancel HW offload. The Application receives the entry pointer upon creation and if can call this function when there is no more need for this offload. For example, if the entry aged, use this API to free it.

__DOCA_EXPERIMENTAL int doca_flow_port_pair (doca_flow_port *port, doca_flow_port *pair_port)

pair two doca flow ports.

Parameters

port
    Pointer to doca flow port.
pair_port
    Pointer to the pair port.

Returns

0 on success, negative on failure.

Description

This API should be used to pair two doca ports. This pair should be the same as the actual physical layer paired information. Those two pair ports have no order, a port cannot be paired with itself.

In this API, default behavior will be handled according to each modes. In VNF mode, pair information will be translated to queue action to redirect packets to it's pair port. In SWITCH and REMOTE_VNF mode, default rules will be created to redirect packets between 2 pair ports.

__DOCA_EXPERIMENTAL void doca_flow_port_pipes_dump (uint16_t port_id, FILE *f)

Dump pipe of one port.

Parameters

port_id
    Port id of the port.
f
    The output file of the pipe information.

Description

Dump all pipes information belong to this port.

__DOCA_EXPERIMENTAL void doca_flow_port_pipes_flush (uint16_t port_id)

Flush pipes of one port.

Parameters

port_id
    Port id of the port.

Description

Destroy all pipes and all pipe entries belonging to the port.

__DOCA_EXPERIMENTAL uint8_t *doca_flow_port_priv_data (doca_flow_port *port)

Get pointer of user private data.

Parameters

port
    Port struct.

Returns

Private data head pointer.

Description

User can manage specific data structure in port structure. The size of the data structure is given on port configuration. See doca_flow_cfg for more details.

__DOCA_EXPERIMENTAL doca_flow_port *doca_flow_port_start (const doca_flow_port_cfg *cfg, doca_flow_error *error)

Start a doca port.

Parameters

cfg
    Port configuration, see doca_flow_cfg for details.
error
    Output error, set doca_flow_error for details.

Returns

Port handler on success, NULL otherwise and error is set.

Description

Start a port with the given configuration. Will create one port in the doca flow layer, allocate all resources used by this port, and create the default offload flows including jump and default RSS for traffic.

__DOCA_EXPERIMENTAL int doca_flow_port_stop (doca_flow_port *port)

Stop a doca port.

Parameters

port
    Port struct.

Returns

0 on success, negative on failure.

Description

Stop the port, disable the traffic.

__DOCA_EXPERIMENTAL int doca_flow_query (doca_flow_pipe_entry *entry, doca_flow_query *query_stats)

Extract information about specific entry.

Parameters

entry
    The pipe entry toe query.
query_stats
    Data retrieved by the query.

Returns

0 on success, negative on failure.

Description

Query the packet statistics about specific pipe entry

__DOCA_EXPERIMENTAL int doca_flow_shared_resource_cfg (doca_flow_shared_resource_type type, uint32_t id, doca_flow_shared_resource_cfg *cfg, doca_flow_error *error)

Configure a single shared resource.

Parameters

type
Shared resource type.
id
Shared resource id.
cfg
Pointer to a shared resource configuration.
error
Output error, set doca_flow_error for details.

Returns

0 on success, negative on failure.

Description

This API can be used by bounded and unbounded resources.

__DOCA_EXPERIMENTAL int doca_flow_shared_resources_bind (doca_flow_shared_resource_type type, uint32_t *res_array, uint32_t res_array_len, void *bindable_obj, doca_flow_error *error)

Binds a bulk of shared resources to a bindable object.

Parameters

type
Shared resource type.

res_array
Array of shared resource IDs.

res_array_len
Shared resource IDs array length.

bindable_obj
Pointer to an allowed bindable object, use NULL to bind globally.

error
Output error, set doca_flow_error for details.

Returns

0 on success, negative on failure.

Description

Binds a bulk of shared resources from the same type to a bindable object. Currently the bindable objects are ports and pipes.

#define DOCA_FLOW_META_MAX 20

Max meta data size in bytes.

2.9.   Flow (grpc)

DOCA flow grpc API to run remote HW offload with flow library. For more details please refer to the user guide on DOCA devzone.

struct doca_flow_grpc_bindable_obj

bindable object configuration

struct doca_flow_grpc_fwd

forwarding configuration wrapper

struct doca_flow_grpc_pipe_cfg

pipeline configuration wrapper

struct doca_flow_grpc_response

General DOCA Flow response struct.

enum doca_flow_grpc_bindable_obj_type

doca flow grpc bindable object types

Values

DOCA_FLOW_GRPC_BIND_TYPE_PIPE
 bind resource to a pipe
DOCA_FLOW_GRPC_BIND_TYPE_PORT
 bind resource to a port
DOCA_FLOW_GRPC_BIND_TYPE_NULL
 bind resource globally

__DOCA_EXPERIMENTAL void doca_flow_grpc_client_create (char *grpc_address)

Initialize a channel to DOCA flow grpc server.

Parameters

grpc_address
 String representing the service ip, i.e. "127.0.0.1" or "192.168.100.3:5050". If no port is provided, it will use the service default port.

Description

Must be invoked first before any other function in this API. this is a one time call, used for grpc channel initialization.

doca_flow_grpc_control_pipe_add_entry (uint16_t pipe_queue, uint8_t priority, uint64_t pipe_id, doca_flow_match *match, doca_flow_match *match_mask, doca_flow_grpc_fwd *client_fwd)

RPC call for doca_flow_control_pipe_add_entry().

Parameters

pipe_queue
Queue identifier.
priority
Priority value..
pipe_id
Pipe ID.
match
Pointer to match, indicate specific packet match information.
match_mask
Pointer to match mask information.
client_fwd
Pointer to fwd actions.

Returns

doca_flow_grpc_response.

doca_flow_grpc_create_pipe (doca_flow_grpc_pipe_cfg *cfg, doca_flow_grpc_fwd *fwd, doca_flow_grpc_fwd *fwd_miss)

RPC call for doca_flow_create_pipe().

Parameters

cfg
Pipe configuration, see doca_flow_grpc_pipe_cfg for details.
fwd
Fwd configuration for the pipe.
fwd_miss
Fwd_miss configuration for the pipe. NULL for no fwd_miss. When creating a pipe if there is a miss and fwd_miss configured, packet steering should jump to it.

Returns

doca_flow_grpc_response.

__DOCA_EXPERIMENTAL void doca_flow_grpc_destroy (void)

RPC call for doca_flow_destroy().

doca_flow_grpc_destroy_pipe (uint16_t port_id, uint64_t pipe_id)

RPC call for doca_flow_destroy_pipe().

Parameters

port_id
Port ID.
pipe_id
Pipe ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_destroy_port (uint16_t port_id)

RPC call for doca_flow_destroy_port().

Parameters

port_id
Port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_entries_process (uint16_t port_id, uint16_t pipe_queue, uint64_t timeout, uint32_t max_processed_entries)

RPC call for doca_flow_grpc_entries_process().

Parameters

port_id
Port ID

pipe_queue
Queue identifier.
timeout
Max time in micro seconds for this function to process entries. Process once if timeout is 0
max_processed_entries
Flow entries number to process If it is 0, it will proceed until timeout.

Returns

doca_flow_grpc_response

doca_flow_grpc_entry_get_status (uint64_t entry_id)

RPC call for doca_flow_entry_get_status().

Parameters

entry_id
pipe entry ID

Returns

doca_flow_grpc_response

doca_flow_grpc_env_destroy (void)

Invoked dpdk destroy.

Returns

doca_flow_grpc_response.

doca_flow_grpc_env_init (application_dpdk_config *dpdk_config)

Invoked dpdk initializations.

Parameters

dpdk_config
pointer to application_dpdk_config, hold some initialization configuration info.

Returns

doca_flow_grpc_response.

Description

Must be invoked before DOCA Flow client API. Initialize DPDK ports, including mempool allocation. Initialize hairpin queues if needed. Binds hairpin queues of each port to its peer port.

doca_flow_grpc_handle_aging (uint16_t port_id, uint16_t queue, uint64_t quota, uint64_t *entries_id, int len)

RPC call for doca_flow_grpc_handle_aging().

Parameters

port_id
Port id to handle aging
queue
Queue identifier.
quota
Max time quota in micro seconds for this function to handle aging.
entries_id
User input entries array for the aged flows.
len
User input length of entries array.

Returns

doca_flow_grpc_response.

doca_flow_grpc_init (doca_flow_cfg *cfg)

RPC call for doca_flow_init().

Parameters

cfg
Program configuration, see doca_flow_cfg for details.

Returns

doca_flow_grpc_response.

doca_flow_grpc_pipe_add_entry (uint16_t pipe_queue, uint64_t pipe_id, doca_flow_match *match, doca_flow_actions *actions, doca_flow_monitor *monitor, doca_flow_grpc_fwd *client_fwd, uint32_t flags)

RPC call for doca_flow_pipe_add_entry().

Parameters

pipe_queue
 Queue identifier.
pipe_id
 Pipe ID.
match
 Pointer to match, indicate specific packet match information.
actions
 Pointer to modify actions, indicate specific modify information.
monitor
 Pointer to monitor actions.
client_fwd
 Pointer to fwd actions.
flags
 Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

Returns

doca_flow_grpc_response.

doca_flow_grpc_pipe_rm_entry (uint16_t pipe_queue, uint64_t entry_id)

RPC call for doca_flow_control_pipe_add_entry().

Parameters

pipe_queue
 Queue identifier.
entry_id
 The entry ID to be removed.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_pair (uint16_t port_id, uint16_t pair_port_id)

RPC call for doca_flow_port_pair().

Parameters

port_id
port ID.
pair_port_id
pair port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_pipes_dump (uint16_t port_id, FILE *f)

RPC call for doca_flow_port_pipes_dump().

Parameters

port_id
Port ID.
f
The output file of the pipe information.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_pipes_flush (uint16_t port_id)

RPC call for doca_flow_port_pipes_flush().

Parameters

port_id
Port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_start (doca_flow_port_cfg *cfg)

RPC call for doca_flow_port_start().

Parameters

cfg
    Port configuration, see doca_flow_port_cfg for details.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_stop (uint16_t port_id)

RPC call for doca_flow_port_stop().

Parameters

port_id
    Port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_query (uint64_t entry_id, doca_flow_query *query_stats)

RPC call for doca_flow_query().

Parameters

entry_id
    The pipe entry ID to query.
query_stats
    Data retrieved by the query.

Returns

doca_flow_grpc_response.

doca_flow_grpc_shared_resource_cfg (doca_flow_shared_resource_type type, uint32_t id, doca_flow_shared_resource_cfg *cfg)

RPC call for doca_flow_shared_resource_cfg().

Parameters

type
Shared resource type.
id
Shared resource id.
cfg
Pointer to a shared resource configuration.

Returns

doca_flow_grpc_response.

doca_flow_grpc_shared_resources_bind (doca_flow_shared_resource_type type, uint32_t *res_array, uint32_t res_array_len, doca_flow_grpc_bindable_obj *bindable_obj_id)

RPC call for doca_flow_shared_resources_bind().

Parameters

type
Shared resource type.
res_array
Array of shared resource IDs.
res_array_len
Shared resource IDs array length.
bindable_obj_id
Pointer to a bindable object ID.

Returns

doca_flow_grpc_response.

2.10. Flow net

DOCA HW offload flow net structure define. For more details please refer to the user guide on DOCA devzone.

struct doca_flow_ip_addr

doca flow ip address

struct doca_flow_tun

doca flow tunnel information

enum doca_flow_ip_type

doca flow ip address type

Values

DOCA_FLOW_ADDR_NONE = 0
　　ip address is not set
DOCA_FLOW_IP4_ADDR = 4
　　ip address is ipv4
DOCA_FLOW_IP6_ADDR = 6
　　ip address is ipv6

enum doca_flow_tun_type

doca flow tunnel type

Values

DOCA_FLOW_TUN_NONE = 0
　　tunnel is not set
DOCA_FLOW_TUN_VXLAN
　　tunnel is vxlan type
DOCA_FLOW_TUN_GTPU
　　tunnel is gtpu type
DOCA_FLOW_TUN_GRE
　　tunnel is gre type

typedef uint16_t doca_be16_t

16-bit big-endian value.

typedef uint32_t doca_be32_t

32-bit big-endian value.

typedef uint64_t doca_be64_t

64-bit big-endian value.

#define DOCA_ETHER_ADDR_LEN (6)

length of ether add length.

#define DOCA_ETHER_TYPE_IPV4 (0x0800)

IPv4 Protocol.

#define DOCA_ETHER_TYPE_IPV6 (0x86DD)

IPv6 Protocol.

#define DOCA_ETHER_TYPE_TEB (0x6558)

Transparent Ethernet Bridging.

#define DOCA_GTPU_PORT (2152)

gtpu upd port id.

#define DOCA_PROTO_GRE (47)

Cisco GRE tunnels (rfc 1701,1702).

#define DOCA_PROTO_TCP (6)

Transmission Control Protocol.

#define DOCA_PROTO_UDP (17)

User Datagram Protocol.

#define DOCA_VXLAN_DEFAULT_PORT (4789)

default vxlan port id.

2.11.   RegEx engine

DOCA RegEx library. For more details please refer to the user guide on DOCA devzone.

struct doca_regex_job_request

struct doca_regex_job_response

struct doca_regex_match

enum doca_regex_status_flag

Response status flags

Values
DOCA_REGEX_STATUS_SEARCH_FAILED = 1

__DOCA_EXPERIMENTAL doca_regex *doca_regex_create (void)

Returns
Non NULL doca_regex object on success, NULL otherwise.

Description
Create a DOCA RegEx instance.

__DOCA_EXPERIMENTAL int doca_regex_dequeue (doca_regex *regex, uint16_t qid, doca_regex_job_response *responses, uint8_t max_results)

Parameters
regex
    The RegEx engine to dequeue data from.

qid

ID of queue to read results from.

responses

A pointer to an array of responses. The RegEx engine will place each dequeued result into this array until either the maximum number of responses are returned or all available responses have been returned. Caller assumes ownership of each response returned this way. The engine takes no care to clear any un-populated element of this array, it is the caller responsibility to ensure they only read as many responses as was indicated in the functions return value.

max_results

Maximum number of results to return. the responses array MUST have capacity for at least this many elements.

Returns

[0..max_results] The number of responses dequeued or a negative posix status code.

Description

Dequeue responses from the RegEx engine.

> **Note:**
>
> This function is thread safe when each queue is read by only one thread. If there is a chance that more than one thread could read from to the same queue then external synchronisation must be provided by the application to prevent that circumstance.

__DOCA_EXPERIMENTAL void doca_regex_destroy (doca_regex *regex)

Parameters

regex

Instance to be destroyed, MUST NOT BE NULL.

Description

Destroy DOCA RegEx instance.

> **Note:**
>
> The application must call doca_regex_stop() before destroying a running instance.

__DOCA_EXPERIMENTAL int doca_regex_enqueue (doca_regex *regex, uint16_t qid, doca_regex_job_request *job, bool allow_aggregation)

Parameters

regex
> The RegEx engine instance to use.

qid
> ID of queue to enqueue the job into.

job
> The job to enqueue. Caller retains ownership of the job

allow_aggregation
> When set the RegEx engine may choose to not begin processing this job immediately in an attempt to maximise overall efficiency and throughput. When not set the RegEx engine must begin processing immediately, potentially reducing latency. This allows an application to favour either throughput or latency. If in doubt it is recommended to favour throughput.

Returns

[0..1] Number of jobs enqueued or a negative posix status code.

Description

Enqueue a job.

> 📝 **Note:**
>
> This function is thread safe when each queue is written to by only one thread. If there is a chance that more than one thread could write to the same queue then external synchronisation must be provided by the application to prevent that circumstance.

__DOCA_EXPERIMENTAL int doca_regex_hw_device_set (doca_regex *regex, doca_regex_device *device)

Parameters

regex
> The RegEx engine instance to use.

device
> Provide a non NULL value to register a device to use for hardware accelerated searches. The application retains ownership of the device object. The Application shall ensure the lifetime of the device is maintained until the device is either de-registered or the doca_regex instance is destroyed. Pass a NULL value to this function to de-register a registered device.

Returns

0 on success or a negative status code on failure.

Description

Set the device to be used for hardware accelerated searches

> 📰 **Note:**
>
> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
>
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_num_qps_set (doca_regex *regex, uint16_t num_qps)

Parameters

regex
> The RegEx engine instance to use.

num_qps
> Number of QP's that can be used. Must be at least 1.

Returns

0 on success or a negative status code on failure.

Description

Specify how many qp's are available for use

> 📰 **Note:**

> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
>
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_overlap_size_set (doca_regex *regex, uint16_t nb_overlap_bytes)

Parameters

regex
    The RegEx engine instance to use.
nb_overlap_bytes
    Number of bytes of overlap to use.

Returns

0 on success or a negative status code on failure.

Description

Set the size of overlap to use when a job exceeds a devices maximum search size.

When a submitted job is larger than the receiving device can support it must be fragmented. This can cause issues if a match exists but is split across two fragments. To remedy this an overlap size can be set so that these matches may be detected. The overlap defined by this function specifies how many bytes of the previous search fragment will be resent as part of the next search fragment. So for example if a 100 byte job is submitted and a device supported a 32 byte maximum job length then the jobs sent would look as follows:

Overlap size First job Second Job Third Job Fourth job Fifth Job Sixth Job 0 [0-31] [32-63] [64-95] [96-99] --- --- 8 [0-31] [24-55] [42-79] [72-99] --- --- 16 [0-31] [16-47] [32-63] [48-79] [64-95] [80-99]

This allows the user to select an overlap value which provides enough overlap to detect any match they must find for the lowest cost.

> 📝 **Note:**
>
> The maximum overlap size is the smallest of the registered devices maximum job length minus one. This is validated during doca_regex_start.

> **Note:**
>
> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
>
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_program_compiled_rules (doca_regex *regex, const char *hw_compiled_rules_bin, size_t hw_compiled_rules_size, const char *sw_compiled_rules_bin, size_t sw_compiled_rules_size)

Parameters

regex
    The RegEx instance to program.
hw_compiled_rules_bin
    Buffer of pre-compiled binary rules data suitable for use by the selected hw impl.
hw_compiled_rules_size
    Number of bytes of data in the buffer pointed to by hw_compiled_rules_bin.
sw_compiled_rules_bin
    Buffer of pre-compiled binary rules data suitable for use by the selected sw impl.
sw_compiled_rules_size
    Number of bytes of data in the buffer pointed to by sw_compiled_rules_bin.

Returns

0 on success of writing each rule to at least one regex_impl, or a negative posix status code.

Description

Program the given rules buffer(s) into the DOCA RegEx engine.

__DOCA_EXPERIMENTAL int doca_regex_program_compiled_rules_file (doca_regex *regex, const char *hw_compiled_rules_path, const char *sw_compiled_rules_path)

Parameters

regex
   The RegEx instance to program.
hw_compiled_rules_path
   Path to a pre-compiled binary rules data suitable for use by the selected hw impl.
sw_compiled_rules_path
   Path to a pre-compiled binary rules data suitable for use by the selected sw impl.

Returns

0 on success of writing each rule to at least one regex_impl, or a negative posix status code.

Description

Program the given rules file(s) into the DOCA RegEx engine.

__DOCA_EXPERIMENTAL int doca_regex_qp_mempool_set (doca_regex *regex, doca_regex_mempool *mp, uint16_t qid)

Parameters

regex
   The RegEx engine instance.
mp
   Provide a non NULL value to register a memory pool. Provide NULL to de-register it.
qid
   ID of the queue to associate this mempool with

Returns

0 on success or a negative status code on failure.

Description

Register a memory pool to the engine so it can acquire doca_regex_match objects without requiring memory allocations. If an application does not desire to get match details from a search then this can be left out. The application retains ownership of the memory pool and must ensure the memory pool remains available until either the doca regex engine is destroyed or it can be de-registered while the engine is stopped.

> **Note:**
> ▶ This mempool will be used to allocate all matches for all jobs on the registered queue so it must be sized sufficiently large.
> ▶ The application must return the matches in each doca_regex_job_response to the mempool once they no longer needed. See doca_regex_mempool_obj_put.

> **Note:**
> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_small_job_sw_offload_threshold_set (doca_regex *regex, uint16_t threshold)

Parameters

regex
   The RegEx engine instance to use.
threshold
   Number of bytes, when a job has less bytes then threshold the engine will prefer to send the job to the software RegEx device.

Returns

0 on success or a negative status code on failure.

Description

Define a threshold for "small jobs" where it is preferable to scan them using a software RegEx device instead of the hardware RegEx device. This can be useful in cases where a

particular hardware device incurs more overhead in the process of sending the job to and from hardware. The value that should be set here will vary depending on the capability of the hardware device in use.

> **Note:**
>
> ► Defaults to 256.
> ► This value is only considered when both software and hardware RegEx devices are configured.

> **Note:**
>
> ► this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
> ► this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_start (doca_regex *regex)

Parameters

regex
   The DOCA RegEx instance to start, MUST NOT be NULL..

Returns

0 on success or a negative posix status code.

Description

Make the RegEx Engine ready for processing jobs.

After calling this functions the engine will not accept any further configuration but will start accepting jobs. If further configuration is required the engine should be stopped first to allow this configuration to be performed.

__DOCA_EXPERIMENTAL int doca_regex_stop (doca_regex *regex)

Parameters

regex
　　The DOCA RegEx instance to stop, MUST NOT be NULL.

Returns

0 on success or a negative posix status code.

Description

Stop the RegEx engine.

Disable processing of jobs and return the engine to an idle state. The engine will then be ready to accept new configuration or be ready to be destroyed. Any in flight jobs when this function is called will be terminated and will not return any results.

__DOCA_EXPERIMENTAL int doca_regex_sw_device_set (doca_regex *regex, doca_regex_device *device)

Parameters

regex
　　The RegEx engine instance to use.
device
　　Provide a non NULL value to register a device to use for software based searches. The application retains ownership of the device object. The Application shall ensure the lifetime of the device is maintained until the device is either de-registered or the doca_regex instance is destroyed. Pass a NULL value to this function to de-register a registered device.

Returns

0 on success or a negative status code on failure.

Description

Set the device to be used for software based searches

> **Note:**

> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
>
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

__DOCA_EXPERIMENTAL int doca_regex_sw_fallback_enabled_set (doca_regex *regex, bool enabled)

Parameters

regex
   The RegEx engine instance to use.
enabled
   Feature is enabled: true = enabled, false = disabled

Returns

0 on success or a negative status code on failure.

Description

Enable or disable software fallback. When a job is unable to be executed on a hardware RegEx device the engine can automatically re-execute the job on the software device.

> 🗎 **Note:**
>
> ▶ Defaults to false (off).
>
> ▶ This value is only considered when both software and hardware RegEx devices are configured.

> 🗎 **Note:**
>
> ▶ this function shall only be called when the doca_regex instance is not running. see doca_regex_start, doca_regex_stop
>
> ▶ this function is not thread safe. Application must ensure that this is only called from a single thread.

2.12.   RegEx engine memory pool

Define functions to allow easy creation and use of memory pools.

__DOCA_EXPERIMENTAL doca_regex_mempool *doca_regex_mempool_create (size_t elem_size, size_t nb_elems)

Parameters

elem_size
   Size of an element to be stored in the memory pool.
nb_elems
   Number of element stored in the memory pool.

Returns

Pointer to the memory pool on success or NULL on failure.

Description

Create a memory pool.

> 📰 **Note:**
>
> Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL void doca_regex_mempool_destroy (doca_regex_mempool *pool)

Parameters

pool
   Memory pool to be destroyed. Must not be NULL.

Description

Destroy a memory pool and all objects it owned.

> 📃 **Note:**
>
> all pointers to elements in this pool must be cleared before this call. Failure to do socmay result in undefined behaviour.

__DOCA_EXPERIMENTAL void *doca_regex_mempool_get_nth_element (doca_regex_mempool *pool, size_t n)

Parameters

pool
Memory pool to fetch an object from.

n
Index of the object to be retrieved

Returns

Pointer to located object when n is a valid index or NULL

Description

Directly access an object in the mempool by index.

> 📃 **Note:**
>
> ▶ this function does not care if the object is in use or free.
>
> ▶ Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL int doca_regex_mempool_index_of (doca_regex_mempool *pool, const void *obj)

Parameters

pool
Memory pool owning the object.

obj

    Object owned by pool for which an index is to be obtained.

Returns

0 based index of element or a negative error code.

Description

Determine the index of a particular element to allow for index based access to the pool.

> 📝 **Note:**
>
> Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL void *doca_regex_mempool_obj_get (doca_regex_mempool *pool)

Parameters

pool

    Pool from which to get a free object.

Returns

Pointer to an object or NULL if the pool is exhausted.

Description

Get an object from the memory pool.

> 📝 **Note:**
>
> Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL void doca_regex_mempool_obj_put (doca_regex_mempool *pool, void *obj)

Parameters

pool
> Pool which created obj.

obj
> Object created by pool which is being returned to the free state.

Description

Put an object back into the memory pool.

> 📨 **Note:**
>
> Supports single producer and single consumer only.

2.13. Telemetry Service Library

DOCA lib for exporting events to the telemetry service.

DOCA lib for exporting a netflow packet to a netflow collector through the telemetry service.

This lib simplifies and centralizes the formatting and exporting of netflow packets. Netflow is a protocol for exporting information about the device network flows to a netflow collector that will aggregate and analyze the data. After creating conf file and invoke init function, the lib send function can be called with netflow struct to send a netflow packet with the format to the collector of choice specified in the conf file. The lib uses the netflow protocol specified by cisco.

See also:

https://netflow.caligare.com/netflow_v9.htm

Limitations:

The lib supports the netflow V9 format. The lib is not thread safe.

struct doca_telemetry_buffer_attr_t

DOCA schema buffer attribute. Applied to all DOCA sources.

struct doca_telemetry_field_info_t

DOCA schema field.

struct doca_telemetry_file_write_attr_t

DOCA schema file write attribute. Applied to all DOCA sources.

struct doca_telemetry_ipc_attr_t

DOCA schema file write attribute. Applied to all DOCA sources.

struct doca_telemetry_ipc_timeout_attr_t

DOCA schema IPC attribute. Applied to all DOCA sources.

struct doca_telemetry_netflow_flowset_field

One field in netflow template, please look at doca_telemetry_netflow_types for type macros.

struct doca_telemetry_netflow_send_attr_t

DOCA netflow sending attribute. Applied to all DOCA sources.

struct doca_telemetry_netflow_template

Template for the records. struct record_example { uint32_t src_addr_V4; uint32_t dst_addr_V4; } struct doca_telemetry_netflow_flowset_field fields[] = { {.type = DOCA_NETFLOW_IPV4_SRC_ADDR, .length = DOCA_NETFLOW_IPV4_SRC_ADDR_DEFAULT_LENGTH}, {.type = DOCA_NETFLOW_IPV4_DST_ADDR, .length = DOCA_NETFLOW_IPV4_DST_ADDR_DEFAULT_LENGTH} }; struct doca_telemetry_netflow_template netflow_template = { .field_count = 2; .fields = fields; };.

struct doca_telemetry_opaque_events_attr_t

DOCA schema opaque events attribute. Applied to all DOCA sources.

struct doca_telemetry_source_name_attr_t

DOCA telemetry source attributes: id and tag.

enum telemetry_status

DOCA telemtry status.

Values

DOCA_TELEMETRY_OK = 0
    ok status
DOCA_TELEMETRY_ERROR = 1
    general error
DOCA_TELEMETRY_ALLOC_ERROR
    memory allocation error
DOCA_TELEMETRY_CLX_CONTEXT_INIT_ERROR
    context init error
DOCA_TELEMETRY_CLX_CONTEXT_CLONE_ERROR
    context clone error
DOCA_TELEMETRY_SOURCE_ATTR_NOT_SET
    attribute not set error
DOCA_TELEMETRY_INTERNAL_BUFFER_ERROR
    buffer internal error
DOCA_TELEMETRY_BAD_STATE_ERROR
    general bad state error
DOCA_TELEMETRY_BAD_PARAM_ERROR
    general bad parameter error

typedef uint8_t doca_guid_t

DOCA GUID type.

typedef uint64_t doca_telemetry_timestamp_t

DOCA schema type index type.

typedef uint8_t doca_telemetry_type_index_t

DOCA schema field type index.

__DOCA_EXPERIMENTAL int doca_telemetry_check_ipc_status (void *doca_source)

Return status of IPC transport.

Parameters

doca_source
    Input doca source.

Returns

1 if IPC is disabled from config. 0 (DOCA_TELEMETRY_OK) if IPC is connected. negative telemetry_status if IPC is not connected. This status occurs after data send_receive

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_buffer_attr_set (doca_telemetry_buffer_attr_t *buffer_attr)

Set buffer attributes to DOCA netflow.

Parameters

buffer_attr
     Attributes to set.

Returns

0 on success, a negative telemetry_status on error.

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_destroy (void)

Free the exporter memory and close the connection.

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_file_write_attr_set (doca_telemetry_file_write_attr_t *file_attr)

Set file write attributes to DOCA netflow.

Parameters

file_attr
     Attributes to set.

Returns

0 on success, a negative telemetry_status on error.

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_flush (void)

Immediately flush the data of the DOCA internal Netflow source.

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_init (uint16_t source_id)

Init exporter memory, set configs and open connection.

Parameters

source_id
  Unique source ID.

Returns

0 on success, a negative telemetry_status on error.

Description

The Source ID field is a 32-bit value that is used to guarantee uniqueness for all flows exported from a particular device (see link).

This function can be called again only after doca_telemetry_netflow_destroy was called.

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_ipc_attr_set (doca_telemetry_ipc_attr_t *ipc_attr)

Set IPC tarnsport attributes to DOCA netflow.

Parameters

ipc_attr
  Attribute to set.

Returns

0 on success, a negative telemetry_status on error.

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_send (const doca_telemetry_netflow_template *netflow_template, const void **records, size_t nof_records, size_t *nof_records_sent)

Sending netflow records. Need to init first.

Parameters

netflow_template

Template pointer of how the records are structured. For more info reffer to doca_telemetry_netflow_template.

records

Array of pointers to the flows structs to send, must be packed. Strings must be an array in the struct, not a pointer.

nof_records

Records array size.

nof_records_sent

If not NULL, it will be filled with amount of records sent.

Returns

0 on success, a negative telemetry_status on error.

Description

> 📝 **Note:**
>
> When sending more then 30 records the lib splits the records to multiple packets because each packet can only send up to 30 records (Netflow protocol limit)

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_send_attr_set (doca_telemetry_netflow_send_attr_t *netflow_send_attr)

Set send attributes to DOCA netflow.

Parameters

netflow_send_attr
   Attribute to set.

Returns

0 on success, a negative telemetry_status on error.

__DOCA_EXPERIMENTAL int doca_telemetry_netflow_start (doca_telemetry_source_name_attr_t *source_attr)

Finalizes netflow setup.

Parameters

source_attr
   Source attributes.

Returns

0 on success, a negative telemetry_status on error.

Description

Do NOT set attributes after this call.

__DOCA_EXPERIMENTAL int doca_telemetry_schema_add_type (void *doca_schema, const char *new_type_name, doca_telemetry_field_info_t *fields, int num_fields, doca_telemetry_type_index_t *type_index)

Add user-defined fields to create new type in DOCA schema.

Parameters

doca_schema
    Schema to create type in.
new_type_name
    Name for new type.
fields
    User-defined fields.
num_fields
    Number of user defined fields.
type_index
    Type index for the created type is written to this variable.

Returns

0 on success, a negative telemetry_status on error

__DOCA_EXPERIMENTAL int doca_telemetry_schema_buffer_attr_set (void *doca_schema, doca_telemetry_buffer_attr_t *buffer_attr)

Set buffer attributes to DOCA schema.

Parameters

doca_schema
    Input schema.
buffer_attr
    Attribute to set.

Returns

0 on success, a negative telemetry_status on error

__DOCA_EXPERIMENTAL void doca_telemetry_schema_destroy (void *doca_schema)

Destructor for DOCA schema.

Parameters

doca_schema
    Schema to destroy.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_file_write_attr_set (void *doca_schema, doca_telemetry_file_write_attr_t *file_attr)

Set file write attributes to DOCA schema.

Parameters

doca_schema
    Input schema.
file_attr
    Attribute to set.

__DOCA_EXPERIMENTAL void *doca_telemetry_schema_init (const char *schema_name)

Initialize DOCA schema to prepare it for setting attributes and adding types. DOCA schema is used to initialize DOCA sources that will collect the data according to the same schema.

Parameters

schema_name
    Name of the schema.

Returns

Pointer to DOCA schema, NULL on error.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_ipc_attr_set (void *doca_schema, doca_telemetry_ipc_attr_t *ipc_attr)

Set IPC tarnsport attributes to DOCA schema.

Parameters

doca_schema
   Input schema.
ipc_attr
   Attribute to set.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_ipc_timeouts_attr_set (void *doca_schema, doca_telemetry_ipc_timeout_attr_t *ipc_timeout_attr)

Set ipc timeout attributes to DOCA schema.

Parameters

doca_schema
   Input schema.
ipc_timeout_attr
   Attribute to set.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_opaque_events_attr_set (void *doca_schema, doca_telemetry_opaque_events_attr_t *opaque_events_attr)

Set Opaque events attributes to DOCA shcema.

Parameters

doca_schema
   Input schema.
opaque_events_attr
   Attribute to set.

__DOCA_EXPERIMENTAL int doca_telemetry_schema_start (void *doca_schema)

Finalizes schema setup to start creating Doca Sources from the schema.

Parameters

doca_schema
  Input schema to start.

Returns

0 on success, a negative telemetry_status on error

Description

Do NOT add new types after this function was called.

__DOCA_EXPERIMENTAL void *doca_telemetry_source_create (void *doca_schema)

Creates a single DOCA source from schema.

Parameters

doca_schema
  Schema from which source will be created.

Returns

pointer to DOCA source, or NULL on error.

Description

To create a DOCA source, first call doca_telemetry_schema_start() to prepare the DOCA schema.

__DOCA_EXPERIMENTAL void doca_telemetry_source_destroy (void *doca_source)

Destructor for DOCA source.

Parameters

doca_source
  Source to destroy.

__DOCA_EXPERIMENTAL void doca_telemetry_source_flush (void *doca_source)

Immediately flush the data of the DOCA source. This function is not thread-safe and should not be called from different threads without proper access control.

Parameters

doca_source
   DOCA source to flush.

__DOCA_EXPERIMENTAL void doca_telemetry_source_name_attr_set (void *doca_source, doca_telemetry_source_name_attr_t *source_attr)

Set source attributes to DOCA.

Parameters

doca_source
   Source to update.
source_attr
   Source attribute to set.

Description

source_tag is set on schema basis while source_id is set on source basis.

__DOCA_EXPERIMENTAL int doca_telemetry_source_opaque_report (void *doca_source, const doca_guid_t app_id, uint64_t user_defined1, uint64_t user_defined2, const void *data, uint32_t data_size)

Report opaque event data via DOCA source.

Parameters

doca_source
   Source to report.

app_id
    User defined application ID.
user_defined1
    User defined parameter 1.
user_defined2
    User defined parameter 2.
data
    Data buffer.
data_size
    Size of the data in the data buffer.

Returns

0 on success, a negative telemetry_status on error.

Description

Data is flushed from internal buffer when the buffer is full. Flushing the data immediately can be done by invoking doca_telemetry_source_flush().

__DOCA_EXPERIMENTAL uint32_t doca_telemetry_source_opaque_report_max_data_size (void *doca_source)

Get max data size for opaque report.

Parameters

doca_source
    Source to report.

Returns

Maximal data size

__DOCA_EXPERIMENTAL int doca_telemetry_source_report (void *doca_source, doca_telemetry_type_index_t index, void *data, int count)

Report events data of the same type via DOCA source.

Parameters

doca_source
   Source to report.
index
   Type index in the DOCA schema.
data
   Data buffer.
count
   Number of events written to the data buffer.

Returns

0 on success, a negative telemetry_status on error

Description

Data is flushed from internal buffer when the buffer is full. Flushing the data immediately can be done by invoking doca_telemetry_source_flush(). This function is not thread-safe and should not be called from different threads without proper access control.

__DOCA_EXPERIMENTAL int doca_telemetry_source_start (void *doca_source)

Applies source attribute and starts DOCA source.

Parameters

doca_source
   DOCA source to start.

Returns

0 on success, a negative telemetry_status on error

Description

Call this function to start reporting.

doca_telemetry_timestamp_t doca_telemetry_timestamp_get (void)

Get timestamp in the proper format.

Returns

Timestamp value.

#define DOCA_GUID_SIZE 16

DOCA GUID size.

#define DOCA_NETFLOW_APP_ID { \ 0x99, 0x10, 0xc1, 0x28, 0x39, 0x61, 0x47, 0xe6,\ 0xbe, 0x6c, 0x71, 0x5a, 0x0f, 0x03, 0xad, 0xd6 \ }

NetFlow Application ID.

> 📨 **Note:**
>
> This GUID cannot change

#define DOCA_TELEMETRY_DEFAULT_BUFFER_SIZE 60000

Default buffer size for the DOCA sources.

#define DOCA_TELEMETRY_DEFAULT_CONFIG_ROOT "/opt/ mellanox/doca/services/telemetry/config"

Default config root folder.

#define DOCA_TELEMETRY_DEFAULT_DATA_ROOT "/ opt/mellanox/doca/services/telemetry/data/"

Default data root folder.

#define DOCA_TELEMETRY_DEFAULT_FILE_AGE 60 * 60 * 1000000L

Default maximal file age.

#define DOCA_TELEMETRY_DEFAULT_FILE_SIZE 1 * 1024 * 1024

Default maximal file size.

#define DOCA_TELEMETRY_DEFAULT_IPC_RECONNECT_RETRIES 3

Default number of IPC reconnection attempts.

#define DOCA_TELEMETRY_DEFAULT_IPC_RECONNECT_TIME 100

Default IPC reconnection time.

#define DOCA_TELEMETRY_DEFAULT_IPC_SOCKET_DIR "/opt/mellanox/doca/services/telemetry/ipc_sockets"

Default IPC socket directory.

#define DOCA_TELEMETRY_DEFAULT_IPC_SOCKET_TIMEOUT 3000

Default IPC socket timeout.

#define DOCA_TELEMETRY_FIELD_TYPE_BOOL "bool"

DOCA_TELEMETRY_FIELD_TYPE_{} are data types that are used to create doca_telemetry_field_info_t;.

DOCA telemtry bool type

#define DOCA_TELEMETRY_FIELD_TYPE_CHAR "char"

DOCA telemtry char type.

#define DOCA_TELEMETRY_FIELD_TYPE_DOUBLE "double"

DOCA telemtry double type.

#define DOCA_TELEMETRY_FIELD_TYPE_FLOAT "float"

DOCA telemtry float type.

#define DOCA_TELEMETRY_FIELD_TYPE_IN "int"

DOCA telemtry in type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT16 "int16_t"

DOCA telemtry int16 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT32 "int32_t"

DOCA telemtry int32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT64 "int64_t"

DOCA telemtry int64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT8 "int8_t"

DOCA telemtry int8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONG "long"

DOCA telemtry long type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONGLONG "long long"

DOCA telemtry longlong type.

#define DOCA_TELEMETRY_FIELD_TYPE_SHORT "short"

DOCA telemtry short type.

#define DOCA_TELEMETRY_FIELD_TYPE_TIMESTAMP DOCA_TELEMETRY_FIELD_TYPE_UINT64

DOCA telemtry timestamp type.

#define DOCA_TELEMETRY_FIELD_TYPE_UCHAR "unsigned char"

DOCA telemtry uchar type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT "unsigned int"

DOCA telemtry uint type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT16 "uint16_t"

DOCA telemtry uint16 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT32 "uint32_t"

DOCA telemtry uint32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT64 "uint64_t"

DOCA telemtry uint64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT8 "uint8_t"

DOCA telemtry uint8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONG "unsigned long"

DOCA telemtry ulong type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONGLONG "long long"

DOCA telemtry ulonglong type.

#define DOCA_TELEMETRY_FIELD_TYPE_USHORT "unsigned short"

DOCA telemtry ushort type.

#define NUM_OF_DOCA_FIELDS (sizeof(type)/ sizeof(doca_telemetry_field_info_t))

NUM_OF_DOCA_FIELDS is macro for fast counting number of fields in user-defined fields array.

Chapter 3.   Data Structures

Here are the data structures with brief descriptions:

doca_argp_param
  Program flag information
doca_argp_program_general_config
  DOCA general flags values as provided to the program
doca_argp_program_type_config
  Information about program configuration
doca_comm_channel_init_attr
  Configuration attributes for endpoint initialization
doca_dma_job_memcpy
doca_dma_memcpy_result
doca_dpi_config_t
  DPI init configuration
doca_dpi_grpc_generic_packet
  Generic packet that holds payload or a whole packet as segment
doca_dpi_grpc_result
  Dequeue result
doca_dpi_parsing_info
  L2-L4 flow information
doca_dpi_result
  Dequeue result
doca_dpi_sig_data
  Extra signature data
doca_dpi_sig_info
  Signature info
doca_dpi_stat_info
  DPI statistics
doca_event
  Activity completion event
doca_flow_action_desc
  Action description
doca_flow_action_descs
  Action descriptions

doca_flow_action_descs_packet
  Packet action descriptions
doca_flow_action_field
  Extended modification action
doca_flow_actions
  Doca flow actions information
doca_flow_aged_query
  Aged flow query callback context
doca_flow_cfg
  Doca flow global configuration
doca_flow_encap_action
  Doca flow encap data information
doca_flow_error
  Doca flow error message struct
doca_flow_fwd
  Forwarding configuration
doca_flow_grpc_bindable_obj
  Bindable object configuration
doca_flow_grpc_fwd
  Forwarding configuration wrapper
doca_flow_grpc_pipe_cfg
  Pipeline configuration wrapper
doca_flow_grpc_response
  General DOCA Flow response struct
doca_flow_ip_addr
  Doca flow ip address
doca_flow_match
  Doca flow matcher information
doca_flow_monitor
  Doca monitor action configuration
doca_flow_pipe_cfg
  Pipeline configuration
doca_flow_port_cfg
  Doca flow port configuration
doca_flow_query
  Flow query result
doca_flow_resource_meter_cfg
  Doca flow meter resource configuration
doca_flow_resources
  Doca flow resource quota
doca_flow_shared_resource_cfg
  Doca flow shared resource configuration

doca_flow_tun
> Doca flow tunnel information

doca_job
> Job structure describes request arguments for service provided by context

doca_regex_buffer

doca_regex_compiled_rules_result

doca_regex_dev_caps

doca_regex_dev_info

doca_regex_device

doca_regex_job_request

doca_regex_job_response

doca_regex_match

doca_telemetry_buffer_attr_t
> DOCA schema buffer attribute. Applied to all DOCA sources

doca_telemetry_field_info_t
> DOCA schema field

doca_telemetry_file_write_attr_t
> DOCA schema file write attribute. Applied to all DOCA sources

doca_telemetry_ipc_attr_t
> DOCA schema file write attribute. Applied to all DOCA sources

doca_telemetry_ipc_timeout_attr_t
> DOCA schema IPC attribute. Applied to all DOCA sources

doca_telemetry_netflow_flowset_field
> One field in netflow template, please look at doca_telemetry_netflow_types for type macros

doca_telemetry_netflow_send_attr_t
> DOCA netflow sending attribute. Applied to all DOCA sources

doca_telemetry_netflow_template
> Template for the records. struct record_example { uint32_t src_addr_V4;
> uint32_t dst_addr_V4; } struct doca_telemetry_netflow_flowset_field
> fields[] = { {.type = DOCA_NETFLOW_IPV4_SRC_ADDR, .length
> = DOCA_NETFLOW_IPV4_SRC_ADDR_DEFAULT_LENGTH},
> {.type = DOCA_NETFLOW_IPV4_DST_ADDR, .length =
> DOCA_NETFLOW_IPV4_DST_ADDR_DEFAULT_LENGTH} }; struct
> doca_telemetry_netflow_template netflow_template = { .field_count = 2; .fields = fields; };

doca_telemetry_opaque_events_attr_t
> DOCA schema opaque events attribute. Applied to all DOCA sources

doca_telemetry_source_name_attr_t
> DOCA telemetry source attributes: id and tag

3.1. doca_argp_param Struct Reference

Program flag information.

> **Note:**
>
> It is the programmer's responsibility to ensure the callback will copy the content of the param passed to it. The pointer pointing to the param is owned by doca_argp, and it is only valid in the scope of the called callback.

enumdoca_argp_type doca_argp_param::arg_type

Flag argument type

char *doca_argp_param::arguments

Flag expected arguments

callback_func doca_argp_param::callback

Flag program callback

char *doca_argp_param::description

Flag description

bool doca_argp_param::is_cli_only

Is flag supported only in cli mode

bool doca_argp_param::is_mandatory

Is flag mandatory for the program

char *doca_argp_param::long_flag

Flag short name

char *doca_argp_param::short_flag

Flag long name

3.2. doca_argp_program_general_config Struct Reference

DOCA general flags values as provided to the program.

char doca_argp_program_general_config::grpc_address

The gRPC server address as provided by the user

int doca_argp_program_general_config::log_level

The log level as provided by the user

3.3. doca_argp_program_type_config Struct Reference

Information about program configuration.

bool doca_argp_program_type_config::is_dpdk

Is the program based on DPDK API

bool doca_argp_program_type_config::is_grpc

Is the program based on gRPC API

3.4. doca_comm_channel_init_attr Struct Reference

Configuration attributes for endpoint initialization.

uint32_t doca_comm_channel_init_attr::cookie

Cookie returned when polling the event_channel.

uint32_t doca_comm_channel_init_attr::flags

Flags: 0 or DOCA_CC_INIT_FLAG_NONBLOCK.

uint32_t doca_comm_channel_init_attr::maxmsgs

Max. # of messages on queue.

uint32_t doca_comm_channel_init_attr::msgsize

Max. message size (bytes).

3.5.   doca_dma_job_memcpy Struct Reference

A job to be dispatched via the DMA library.

struct doca_job doca_dma_job_memcpy::base

Common job data

doca_buf *doca_dma_job_memcpy::dst_buff

Destination data buffer

uint64_t doca_dma_job_memcpy::num_bytes_to_copy

Number of bytes to copy

const doca_buf *doca_dma_job_memcpy::src_buff

Source data buffer

3.6.   doca_dma_memcpy_result Struct Reference

Result of a DMA Memcpy job. Will be held inside the doca_event::result field.

doca_error_t doca_dma_memcpy_result::result

Operation result

3.7.　doca_dpi_config_t Struct Reference

DPI init configuration.

uint32_t doca_dpi_config_t::max_packets_per_queue

Number of packets concurrently processed by the DPI engine.

uint32_t doca_dpi_config_t::max_sig_match_len

The minimum required overlap between two packets for regex match

uint16_t doca_dpi_config_t::nb_queues

Number of DPI queues

3.8.　doca_dpi_grpc_generic_packet Struct Reference

Generic packet that holds payload or a whole packet as segment.

uint16_t doca_dpi_grpc_generic_packet::seg_len

The length of the data inside segment buffer

uint8_t *doca_dpi_grpc_generic_packet::segment

The buffer with data to be scanned by the DPI

3.9.　doca_dpi_grpc_result Struct Reference

Dequeue result.

struct doca_dpi_sig_info doca_dpi_grpc_result::info

Signature information

bool doca_dpi_grpc_result::matched

Indicates flow was matched

doca_dpi_grpc_generic_packet *doca_dpi_grpc_result::pkt

Pkt provided on enqueue

int doca_dpi_grpc_result::status_flags

doca_dpi_flow_status flags

void *doca_dpi_grpc_result::user_data

User data provided on enqueue

3.10. doca_dpi_parsing_info Struct Reference

L2-L4 flow information.

doca_dpi_parsing_info::@6 doca_dpi_parsing_info::dst_ip

IP destination address

doca_dpi_parsing_info::@4 doca_dpi_parsing_info::dst_ip

IP destination address

__be16 doca_dpi_parsing_info::ethertype

Ethertype of the packet in network byte order

Ethertype of the packet in network byte order

in_addr doca_dpi_parsing_info::ipv4

Ipv4 destination address in network byte order

Ipv4 source address in network byte order

in6_addr doca_dpi_parsing_info::ipv6

Ipv6 destination address in network byte order

Ipv6 source address in network byte order

in_port_t doca_dpi_parsing_info::l4_dport

Layer 4 destination port in network byte order

Layer 4 destination port in network byte order

uint8_t doca_dpi_parsing_info::l4_protocol

Layer 4 protocol

in_port_t doca_dpi_parsing_info::l4_sport

Layer 4 source port in network byte order

Layer 4 source port in network byte order

doca_dpi_parsing_info::@7 doca_dpi_parsing_info::src_ip

IP source address

doca_dpi_parsing_info::@5 doca_dpi_parsing_info::src_ip

IP source address

3.11. doca_dpi_result Struct Reference

Dequeue result.

struct doca_dpi_sig_info doca_dpi_result::info

Signature information

bool doca_dpi_result::matched

Indicates flow was matched

rte_mbuf *doca_dpi_result::pkt

Pkt provided on enqueue

int doca_dpi_result::status_flags

doca_dpi_flow_status flags

void *doca_dpi_result::user_data

User data provided on enqueue

3.12.　doca_dpi_sig_data Struct Reference

Extra signature data.

char doca_dpi_sig_data::name

Signature name

uint32_t doca_dpi_sig_data::sig_id

Signature ID as provided in the signature

3.13.　doca_dpi_sig_info Struct Reference

Signature info.

int doca_dpi_sig_info::action

The action as provided in the signature

uint32_t doca_dpi_sig_info::sig_id

Signature ID as provided in the signature

3.14.   doca_dpi_stat_info Struct Reference

DPI statistics.

uint32_t doca_dpi_stat_info::nb_http_parser_based

Total number of http signature matches

uint32_t doca_dpi_stat_info::nb_matches

Total number of signature matches

uint32_t doca_dpi_stat_info::nb_other_l4

Total number of other l4 signature matches

uint32_t doca_dpi_stat_info::nb_other_l7

Total number of other l7 signature matches

uint32_t doca_dpi_stat_info::nb_scanned_pkts

Total number of scanned packets

uint32_t doca_dpi_stat_info::nb_ssl_parser_based

Total number of ssl signature matches

uint32_t doca_dpi_stat_info::nb_tcp_based

Total number of tcp signature matches

uint32_t doca_dpi_stat_info::nb_udp_based

Total number of udp signature matches

3.15. doca_event Struct Reference

Activity completion event.

Event structure defines activity completion of: 1. Completion event of submitted job. 2. CTX received event as a result of some external activity.

doca_data doca_event::result

Event result defined per action type arguments. If the result is as small as 64 bit (E.g., status or similar), it can be accessed as result.u64. Otherwise the data is pointed to by result.ptr, where the size is fixed for each action type.

int doca_event::type

The type of the event originating activity.

doca_data doca_event::user_data

Defines the origin of the given event. For events originating from submitted jobs, this will hold the same user_data provided as part of the job. For events originating from external activity, refer to the documentation of the specific event type.

3.16. doca_flow_action_desc Struct Reference

action description

enumdoca_flow_action_type doca_flow_action_desc::type

type

3.17. doca_flow_action_descs Struct Reference

action descriptions

struct doca_flow_action_descs_packet doca_flow_action_descs::inner

action descriptions of outer packet.

struct doca_flow_action_desc doca_flow_action_descs::meta

action description of meta data.

struct doca_flow_action_descs_packet doca_flow_action_descs::outer

action descriptions of outer packet.

struct doca_flow_action_desc doca_flow_action_descs::tunnel

action description of tunnel.

3.18. doca_flow_action_descs_packet Struct Reference

packet action descriptions

struct doca_flow_action_desc doca_flow_action_descs_packet::dst_ip

action description of destination IP.

struct doca_flow_action_desc doca_flow_action_descs_packet::dst_mac

action description of destination MAC.

struct doca_flow_action_desc doca_flow_action_descs_packet::dst_port

action description of destination L4 port.

struct doca_flow_action_desc doca_flow_action_descs_packet::eth_type

action description of ether type.

struct doca_flow_action_desc doca_flow_action_descs_packet::src_ip

action description of source IP.

struct doca_flow_action_desc doca_flow_action_descs_packet::src_mac

action description of source MAC.

struct doca_flow_action_desc doca_flow_action_descs_packet::src_port

action description of source L4 port.

struct doca_flow_action_desc doca_flow_action_descs_packet::ttl

action description of IPv4 TTL.

struct doca_flow_action_desc doca_flow_action_descs_packet::vlan

action description of VLAN.

3.19. doca_flow_action_field Struct Reference

extended modification action

void *doca_flow_action_field::address

Field address of pipe match to decide field type and byte offset.

uint32_t doca_flow_action_field::offset

Target bit in field from the address.

3.20. doca_flow_actions Struct Reference

doca flow actions information

bool doca_flow_actions::dec_ttl

decrease TTL value

bool doca_flow_actions::decap

when true, will do decap

struct doca_flow_encap_action doca_flow_actions::encap

encap data information

uint32_t doca_flow_actions::flags

action flags

bool doca_flow_actions::has_encap

when true, will do encap

doca_flow_meta doca_flow_actions::meta

modify meta data, pipe action as mask

struct doca_flow_ip_addr doca_flow_actions::mod_dst_ip

modify destination ip address

uint8_t doca_flow_actions::mod_dst_mac

modify destination mac address

doca_be16_t doca_flow_actions::mod_dst_port

modify layer 4 destination port

struct doca_flow_ip_addr doca_flow_actions::mod_src_ip

modify source ip address

uint8_t doca_flow_actions::mod_src_mac

modify source mac address

doca_be16_t doca_flow_actions::mod_src_port

modify layer 4 source port

3.21.   doca_flow_aged_query Struct Reference

aged flow query callback context

uint64_t doca_flow_aged_query::user_data

The user input context, otherwish the doca_flow_pipe_entry pointer

3.22.   doca_flow_cfg Struct Reference

doca flow global configuration

bool doca_flow_cfg::aging

when true, aging is handled by doca

doca_flow_entry_process_cb doca_flow_cfg::cb

callback for entry create/destroy

const char *doca_flow_cfg::mode_args

set doca flow architecture mode switch, vnf

uint32_t doca_flow_cfg::nr_shared_resources

total shared resource per type

uint32_t doca_flow_cfg::queue_depth

Number of pre-configured queue_size, default to 128

uint16_t doca_flow_cfg::queues

queue id for each offload thread

struct doca_flow_resources doca_flow_cfg::resource

resource quota

3.23. doca_flow_encap_action Struct Reference

doca flow encap data information

struct doca_flow_ip_addr doca_flow_encap_action::dst_ip

destination ip address

uint8_t doca_flow_encap_action::dst_mac

destination mac address

struct doca_flow_ip_addr
doca_flow_encap_action::src_ip

source ip address

uint8_t doca_flow_encap_action::src_mac

source mac address

struct doca_flow_tun doca_flow_encap_action::tun

tunnel info

3.24.   doca_flow_error Struct Reference

doca flow error message struct

const char *doca_flow_error::message

Human-readable error message

enumdoca_flow_error_type doca_flow_error::type

Cause field and error types

3.25.   doca_flow_fwd Struct Reference

forwarding configuration

doca_flow_pipe *doca_flow_fwd::next_pipe

next pipe pointer

int doca_flow_fwd::num_of_queues

number of queues

uint16_t doca_flow_fwd::port_id

destination port id

uint32_t doca_flow_fwd::rss_flags

rss offload types

uint32_t doca_flow_fwd::rss_mark

markid of each queues

uint16_t *doca_flow_fwd::rss_queues

rss queues array

enumdoca_flow_fwd_type doca_flow_fwd::type

indicate the forwarding type

3.26. doca_flow_grpc_bindable_obj Struct Reference

bindable object configuration

uint64_t doca_flow_grpc_bindable_obj::pipe_id

pipe id if type is pipe

uint32_t doca_flow_grpc_bindable_obj::port_id

port id if type is port

enumdoca_flow_grpc_bindable_obj_type doca_flow_grpc_bindable_obj::type

bindable object type

3.27. doca_flow_grpc_fwd Struct Reference

forwarding configuration wrapper

doca_flow_fwd *doca_flow_grpc_fwd::fwd

doca flow fwd struct

uint64_t doca_flow_grpc_fwd::next_pipe_id

next pipe id

3.28. doca_flow_grpc_pipe_cfg Struct Reference

pipeline configuration wrapper

doca_flow_pipe_cfg *doca_flow_grpc_pipe_cfg::cfg

doca_flow_pipe_cfg struct

uint16_t doca_flow_grpc_pipe_cfg::port_id

port id

3.29. doca_flow_grpc_response Struct Reference

General DOCA Flow response struct.

uint32_t doca_flow_grpc_response::aging_res

return value from handle aging

uint64_t doca_flow_grpc_response::entry_id

entry id

enumdoca_flow_entry_status doca_flow_grpc_response::entry_status

return value of entry get status

struct doca_flow_error doca_flow_grpc_response::error

Otherwise, this field contains the error information

uint64_t doca_flow_grpc_response::nb_entries_processed

return value from entries process

uint64_t doca_flow_grpc_response::pipe_id

pipe id

bool doca_flow_grpc_response::success

in case of success should be true

3.30. doca_flow_ip_addr Struct Reference

doca flow ip address

doca_be32_t doca_flow_ip_addr::ipv4_addr

ipv4 address if type is ipv4

doca_be32_t doca_flow_ip_addr::ipv6_addr

ipv6 address if type is ipv6

uint8_t doca_flow_ip_addr::type

ip address type

3.31. doca_flow_match Struct Reference

doca flow matcher information

uint32_t doca_flow_match::flags

match items which are no value

struct doca_flow_ip_addr doca_flow_match::in_dst_ip

inner destination ip address if tunnel is used

uint8_t doca_flow_match::in_dst_mac

inner destination mac address

doca_be16_t doca_flow_match::in_dst_port

inner layer 4 destination port if tunnel is used

doca_be16_t doca_flow_match::in_eth_type

inner Ethernet layer type

uint8_t doca_flow_match::in_l4_type

inner layer 4 protocol type if tunnel is used

struct doca_flow_ip_addr doca_flow_match::in_src_ip

inner source ip address if tunnel is used

uint8_t doca_flow_match::in_src_mac

inner source mac address

doca_be16_t doca_flow_match::in_src_port

inner layer 4 source port if tunnel is used

uint8_t doca_flow_match::in_tcp_flags

inner tcp flags

doca_be16_t doca_flow_match::in_vlan_id

inner vlan id

doca_flow_meta doca_flow_match::meta

Programmable meta data.

struct doca_flow_ip_addr doca_flow_match::out_dst_ip

outer destination ip address

uint8_t doca_flow_match::out_dst_mac

outer destination mac address

doca_be16_t doca_flow_match::out_dst_port

outer layer 4 destination port

doca_be16_t doca_flow_match::out_eth_type

outer Ethernet layer type

uint8_t doca_flow_match::out_l4_type

outer layer 4 protocol type

struct doca_flow_ip_addr doca_flow_match::out_src_ip

outer source ip address

uint8_t doca_flow_match::out_src_mac

outer source mac address

doca_be16_t doca_flow_match::out_src_port

outer layer 4 source port

uint8_t doca_flow_match::out_tcp_flags

outer tcp flags

doca_be16_t doca_flow_match::out_vlan_id

outer vlan id

struct doca_flow_tun doca_flow_match::tun

tunnel info

3.32. doca_flow_monitor Struct Reference

doca monitor action configuration

uint32_t doca_flow_monitor::aging

aging time in seconds.

uint64_t doca_flow_monitor::cbs

Committed Burst Size (bytes).

uint64_t doca_flow_monitor::cir

Committed Information Rate (bytes/second).

uint8_t doca_flow_monitor::flags

indicate which actions be included

uint64_t doca_flow_monitor::user_data

aging user data input.

3.33. doca_flow_pipe_cfg Struct Reference

pipeline configuration

doca_flow_action_descs *doca_flow_pipe_cfg::action_descs

action descriptions

doca_flow_actions *doca_flow_pipe_cfg::actions

actions for the pipeline

bool doca_flow_pipe_cfg::is_root

pipeline is root or not

doca_flow_match *doca_flow_pipe_cfg::match

matcher for the pipeline

doca_flow_match *doca_flow_pipe_cfg::match_mask

match mask for the pipeline

doca_flow_monitor *doca_flow_pipe_cfg::monitor

monitor for the pipeline

const char *doca_flow_pipe_cfg::name

name for the pipeline

uint32_t doca_flow_pipe_cfg::nb_flows

Maximum number of flow rules, default is 8k if not set

doca_flow_port *doca_flow_pipe_cfg::port

port for the pipeline

enumdoca_flow_pipe_type doca_flow_pipe_cfg::type

type of pipe. enum doca_flow_pipe_type

3.34. doca_flow_port_cfg Struct Reference

doca flow port configuration

const char *doca_flow_port_cfg::devargs

specific per port type cfg

uint16_t doca_flow_port_cfg::port_id

dpdk port id

uint16_t doca_flow_port_cfg::priv_data_size

user private data

enumdoca_flow_port_type doca_flow_port_cfg::type

mapping type of port

3.35. doca_flow_query Struct Reference

flow query result

uint64_t doca_flow_query::total_bytes

total bytes hit this flow

uint64_t doca_flow_query::total_pkts

total packets hit this flow

3.36. doca_flow_resource_meter_cfg Struct Reference

doca flow meter resource configuration

uint64_t doca_flow_resource_meter_cfg::cbs

Committed Burst Size (bytes).

uint64_t doca_flow_resource_meter_cfg::cir

Committed Information Rate (bytes/second).

3.37. doca_flow_resources Struct Reference

doca flow resource quota

uint32_t doca_flow_resources::nb_counters

Number of counters to configure

uint32_t doca_flow_resources::nb_meters

Number of traffic meters to configure

3.38. doca_flow_shared_resource_cfg Struct Reference

doca flow shared resource configuration

3.39. doca_flow_tun Struct Reference

doca flow tunnel information

doca_be32_t doca_flow_tun::gre_key

gre key

doca_be32_t doca_flow_tun::gtp_teid

gtp teid

doca_be16_t doca_flow_tun::protocol

next protocol

enumdoca_flow_tun_type doca_flow_tun::type

tunnel type

doca_be32_t doca_flow_tun::vxlan_tun_id

vxlan vni(24) + reserved (8).

3.40.　doca_job Struct Reference

Job structure describes request arguments for service provided by context.

A context of given type may serve one or more request types defined as action type (see definition of enum doca_action_type).

DOCA Job layout

SDK job --> +-------------------------+ | DOCA Job (base) | | type | | flags | | ctx | | user data | | | +-----------+------------+ <-- job arguments | | variable size | arguments | SDK specific | . | structure | . | | . | | . | | . | | . | | . | | | +-----------+------------+

doca_ctx *doca_job::ctx

Doca CTX targeted by the job.

int doca_job::flags

Job submission flags (see `enum doca_job_flags`).

int doca_job::type

Defines the type of the job.

doca_data doca_job::user_data

Job identifier provided by user. Will be returned back on completion.

3.41.　doca_regex_buffer Struct Reference

DOCA RegEx buffer supporting different payload types.

All methods of this API provide assurances similar to the strong exception guarantee in that:

▶ A function will complete in its entirety or not at all. They will NEVER leave an argument or the operated upon object in an undetermined state.

▶ In the case of failure the operated upon object and any arguments are not modified in any way and it is as if the function call was never executed.

▶ A failed function call may have written to a log source. This is not considered a change in any objects state. A type to represent a buffer to be scanned by the regex engine.

const void *doca_regex_buffer::address

Pointer to job data, must remain valid until the response for this job is returned

uint32_t doca_regex_buffer::has_mkey

Is there an mkey associated with this memory. 1 = yes, 0 = no. All other values are invalid.

uint32_t doca_regex_buffer::length

Number of bytes of data in this job.

uint32_t doca_regex_buffer::mkey

Mkey value. Only valid when has_mkey is set.

3.42.   doca_regex_compiled_rules_result Struct Reference

Structure describing the result of compilation of rules into a given regex devices preferred format.

void *doca_regex_compiled_rules_result::compiled_rules_data

Pointer to compiled binary rules data.

uint32_t doca_regex_compiled_rules_result::compiled_rules_size

size of compiled binary rules data.

uint32_t *doca_regex_compiled_rules_result::failure_indices

Array of rule IDs of rules that failed to compile. length == num_failures.

uint32_t doca_regex_compiled_rules_result::num_failures

Number of rules that failed to compile.

uint32_t doca_regex_compiled_rules_result::num_successful

Number of rules that successfully compiled.

3.43. doca_regex_dev_caps Struct Reference

Definition of an abstract implementation to be used by doca regex, Provides a separation between implementation of regex routines for the higher level functionality of the doca regex api. Structure describing the capabilities supported by a regex impl.

uint8_t doca_regex_dev_caps::multi_client_supported

Device has multi-client support, 0 if not supported.

uint8_t doca_regex_dev_caps::pmi_supported

Device has PMI support, 0 if not supported.

3.44. doca_regex_dev_info Struct Reference

Structure describing a regex impl.

uint32_t doca_regex_dev_info::max_job_size

Maximum size of a job payload in bytes for a single job.

const char *doca_regex_dev_info::name

A name describing this device.

uint16_t doca_regex_dev_info::version

version of the regex device [MAJOR][MINOR][PATCH].

3.45. doca_regex_device Struct Reference

Structure containing all the functionality required of a regex device by the DOCA regex.

Allows each DOCA regex instance to be supported by any regex_impl. Such as Hyperscan, DPDK regex, or any other regex library that can be adapted to this interface.

(*cleanup_fn) (doca_regex_device *inst)

Cleanup all resources acquired by the regex_impl from either one time setup or during runtime operation.

Parameters

inst
    Instance of the impl.

Returns

0 on success or a negative posix status code.

(*compile_rules_fn) (doca_regex_device *inst, char const *const *raw_rules, uint32_t num_lines)

Compile given raw rules.

Parameters

inst
    Instance of the impl.
const *raw_rules
num_lines
    Number of lines of text to parse.

Returns

A doca_regex_compiled_rules_result describing the number of failed rules and outlining the index of each rule that failed. Caller assumes ownership of this object.

(*configure_nb_qps_fn) (doca_regex_device *inst, uint32_t num_qps)

Configure the number of QP's required by the regex lib from the device.

Parameters

inst
Instance of the impl.

num_qps
Required number of qps.

Returns

0 on success or a negative posix status code.

void *doca_regex_device::cookie

A place to bind any required extra data into the impl.

(*dequeue_jobs_fn) (doca_regex_device *inst, uint16_t qpid, doca_regex_job_response *responses, uint8_t max_results, doca_regex_mempool *matches_mp)

Fetch job responses from the regex_impl.

Parameters

inst
Instance of the impl.

qpid

responses
A pointer to an array of responses. The regex engine will place each dequeued result into this array until either the maximum number of responses are returned or all available responses have been returned

max_results

matches_mp
To get matches back from the regex search provide a non null memory pool which will return items which are least as big as a doca_regex_match. Provide A NULL pool to just get a summary of the number of matches detected.

Returns

[0..max_results] Number of jobs dequeued or a negative posix status code.

(*destroy_fn) (doca_regex_device *inst)

Function suitable to destroy this doca_regex_device.

Parameters

inst
    Instance of the impl.

Set this to NULL to indicate that the engine should not attempt to destroy this object when the engine is destroyed.

(*enqueue_job_fn) (doca_regex_device *inst, uint16_t qpid, doca_regex_job_request const *job, uint64_t internal_job_id, bool allow_aggregation)

Send a job to the regex engine to be processed.

Parameters

inst
    Instance of the impl.
qpid
    ID of qp to use
job
    Details of the job. Caller retains ownership of these objects;
internal_job_id
    Internal job id - this value should be returned by the device when dequeuing the result for this piece. The job request id is the id of the overall job which may have been broken into multiple pieces.
allow_aggregation
    Give permission to the device to hold the job to potentially improve throughput. If false then the device must immediately send the job.

Returns

[0..count] Number of jobs enqueued or a negative posix status code.

(*init_fn) (doca_regex_device *inst, const char *dev_addr)

Do any one time setup work that may be required before performing regex operations.

Parameters

inst
Instance of the impl.
dev_addr
PCIe address of regex device (NULL if not a HW device).

Returns

0 on success or a negative posix status code.

void *doca_regex_device::mem_regs

Memory regions associated with the device (only applicable to HW dev).

void *doca_regex_device::pd

Protection Domain assigned to device (only applicable to HW dev).

(*program_compiled_rules_fn) (doca_regex_device *inst, void const *data, uint32_t data_len)

Program regex device with the supplied compiled rules data.

Parameters

inst
Instance of the impl.
data
A blob of memory in the same format that would be produced by compile_rules_fn. However it may have been read from a pre-compiled input file and not necessarily compiled by this regex_impl. It will be the applications responsibility to ensure that when providing a path to a pre-compiled data within doca_regex_rules_cfg that the file contains data in a format suitable for the selected regex_impl.
data_len
Number of bytes in the blob pointed to by data.

Returns

0 on success or a negative posix status code.

(*query_dev_caps_fn) (doca_regex_device *inst, doca_regex_dev_caps *caps)

Query capabilities of the regex device.

Parameters

inst
    Instance of the impl.
caps
    Capabilities structure to populate. MUST NOT BE NULL.

Returns

0 on success or a negative posix status code.

(*query_dev_info_fn) (doca_regex_device *inst, doca_regex_dev_info *info)

Query information about the regex device.

Parameters

inst
    Instance of the impl.
info
    Informational structure to populate. MUST NOT BE NULL.

Returns

0 on success or a negative posix status code.

3.46. doca_regex_job_request Struct Reference

Data required to dispatch a job to a RegEx engine.

doca_regex_buffer *doca_regex_job_request::buffer

Data for the job

uint64_t doca_regex_job_request::id

ID of the job, useful to correlate the response.

uint16_t doca_regex_job_request::rule_group_ids

IDs which can be used to select which group of rules are used to process this job. Set each value to a non zero value to enable this feature or 0 to ignore it.

3.47.　doca_regex_job_response Struct Reference

Result of a RegEx search

uint32_t doca_regex_job_response::detected_matches

Total number of detected matches.

uint64_t doca_regex_job_response::id

ID of the enqueued job.

doca_regex_match *doca_regex_job_response::matches

Returned matches. Contains num_matches elements as a linked list

uint32_t doca_regex_job_response::num_matches

Total number of returned matches.

uint64_t doca_regex_job_response::status_flags

Response flags. A bit masked field for zero or more status flags. See doca_regex_status_flag

3.48.　doca_regex_match Struct Reference

Description of a RegEx match

uint32_t doca_regex_match::length

Length of matched value.

uint32_t doca_regex_match::match_start

Index relative to the start of the job / stream where the match begins

doca_regex_match *doca_regex_match::next

Allows matches to be linked together for easy management and iteration

uint32_t doca_regex_match::rule_id

ID of rule used to generate this match.

3.49. doca_telemetry_buffer_attr_t Struct Reference

DOCA schema buffer attribute. Applied to all DOCA sources.

Use to set internal buffer_size. All DOCA sources will have buffers of the same size. The buffer is flushed once it is full, or upon invocation of doca_telemetry_source_flush(). The buffer size is set to 60,000 by default. data_root is the data folder for storing the data and data schema_{hash}.json files.

uint64_t doca_telemetry_buffer_attr_t::buffer_size

Size of the internal buffer.

char *doca_telemetry_buffer_attr_t::data_root

Path to a folder where the data and schema will be stored.

3.50. doca_telemetry_field_info_t Struct Reference

DOCA schema field.

uint16_t doca_telemetry_field_info_t::array_length

Array length for this field type. Set to: 1 to register single value or >1 to register array of values.

const char *doca_telemetry_field_info_t::description

Field descripion

const char *doca_telemetry_field_info_t::field_name

Name of field

const char *doca_telemetry_field_info_t::type_name

Name of type that is already in schema (including built-in types).

3.51. doca_telemetry_file_write_attr_t Struct Reference

DOCA schema file write attribute. Applied to all DOCA sources.

Use to enable/disable file write onto storage under data_root. File write is disabled by default.

bool doca_telemetry_file_write_attr_t::file_write_enabled

User defined switch for enabling/disabling local file write. Disabled by the default.

doca_telemetry_timestamp_t doca_telemetry_file_write_attr_t::max_file_age

Maximum file age. Once current file is older than this threshold a new file will be created.

size_t doca_telemetry_file_write_attr_t::max_file_size

Maximum size of binary data file. Once this size is reached, a new binary file will be created.

3.52.   doca_telemetry_ipc_attr_t Struct Reference

DOCA schema file write attribute. Applied to all DOCA sources.

Use to enable/disable ipc transport. Enabled by default. Default ipc_sockets_dir is '/opt/mellanox/doca/services/telemetry/ipc_sockets'

bool doca_telemetry_ipc_attr_t::ipc_enabled

User defined switch for enabling/disabling IPC transport.

char *doca_telemetry_ipc_attr_t::ipc_sockets_dir

Path to a folder containing Telemetry Service sockets.

3.53.   doca_telemetry_ipc_timeout_attr_t Struct Reference

DOCA schema IPC attribute. Applied to all DOCA sources.

Used to overwrite default values of timeouts for attach/reattach attempts and IPC socket timeout.

uint32_t doca_telemetry_ipc_timeout_attr_t::ipc_max_reconnect_time_

Time limit for reconnect attempts. If the limit is reached, the client is considered disconnected. Default is 100 msec.

int doca_telemetry_ipc_timeout_attr_t::ipc_max_reconnect_tries

Number of reconnect attempts during reconnection period. Default is 3.

uint32_t doca_telemetry_ipc_timeout_attr_t::ipc_socket_timeout_msec

Timeout for IPC messaging socket. If timeout is reached during send_receive, client is considered disconnected. Default is 500 msec.

3.54. doca_telemetry_netflow_flowset_field Struct Reference

One field in netflow template, please look at doca_telemetry_netflow_types for type macros.

uint16_t doca_telemetry_netflow_flowset_field::length

field len in bytes (see link)

uint16_t doca_telemetry_netflow_flowset_field::type

field number id (see link)

3.55. doca_telemetry_netflow_send_attr_t Struct Reference

DOCA netflow sending attribute. Applied to all DOCA sources.

Use to enable/disable netflow sending to a collector. Disabled by the defalt.

char *doca_telemetry_netflow_send_attr_t::netflow_collector_addr

User defined netflow collector's IP address

uint16_t doca_telemetry_netflow_send_attr_t::netflow_collector_port

User defined netflow collector's port

3.56. doca_telemetry_netflow_template Struct Reference

Template for the records. struct record_example { uint32_t src_addr_V4; uint32_t dst_addr_V4; } struct doca_telemetry_netflow_flowset_field fields[] = { {.type = DOCA_NETFLOW_IPV4_SRC_ADDR, .length = DOCA_NETFLOW_IPV4_SRC_ADDR_DEFAULT_LENGTH}, {.type =

DOCA_NETFLOW_IPV4_DST_ADDR, .length =
DOCA_NETFLOW_IPV4_DST_ADDR_DEFAULT_LENGTH} }; struct
doca_telemetry_netflow_template netflow_template = { .field_count = 2; .fields = fields; };.

> 📑 **Note:**
>
> all fields are in network byte order.

uint16_t doca_telemetry_netflow_template::field_count

number of fields in 'fields' array

doca_telemetry_netflow_flowset_field *doca_telemetry_netflow_template::fields

array of field info

3.57. doca_telemetry_opaque_events_attr_t Struct Reference

DOCA schema opaque events attribute. Applied to all DOCA sources.

Use to enable/disable opaque events transport. Disabled by default.

bool doca_telemetry_opaque_events_attr_t::opaque_events_enable

User defined switch for enabling/disabling Opaque Events sending.

3.58. doca_telemetry_source_name_attr_t Struct Reference

DOCA telemetry source attributes: id and tag.

ID and Tag are used to create proper folder structure. All the data collected from
the same host is written to "source_id" folder under data root. Binary file will have
{source_tag}_{timestamp}.bin name format.

char *doca_telemetry_source_name_attr_t::source_id

Hostname or guid.

char *doca_telemetry_source_name_attr_t::source_tag

User defined datafile name prefix.

Chapter 4.   Data Fields

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

A

action
   doca_dpi_sig_info
action_descs
   doca_flow_pipe_cfg
actions
   doca_flow_pipe_cfg
address
   doca_flow_action_field
   doca_regex_buffer
aging
   doca_flow_monitor
   doca_flow_cfg
aging_res
   doca_flow_grpc_response
arg_type
   doca_argp_param
arguments
   doca_argp_param
array_length
   doca_telemetry_field_info_t

B

base
   doca_dma_job_memcpy
buffer
   doca_regex_job_request
buffer_size
   doca_telemetry_buffer_attr_t

C

callback
doca_argp_param
cb
doca_flow_cfg
cbs
doca_flow_monitor
doca_flow_resource_meter_cfg
cfg
doca_flow_grpc_pipe_cfg
cir
doca_flow_resource_meter_cfg
doca_flow_monitor
cleanup_fn
doca_regex_device
compile_rules_fn
doca_regex_device
compiled_rules_data
doca_regex_compiled_rules_result
compiled_rules_size
doca_regex_compiled_rules_result
configure_nb_qps_fn
doca_regex_device
cookie
doca_comm_channel_init_attr
doca_regex_device
ctx
doca_job

D

data_root
doca_telemetry_buffer_attr_t
dec_ttl
doca_flow_actions
decap
doca_flow_actions
dequeue_jobs_fn
doca_regex_device
description
doca_argp_param
doca_telemetry_field_info_t

destroy_fn

detected_matches

devargs

dst_buff

dst_ip

dst_mac

dst_port

E

encap

enqueue_job_fn

entry_id

entry_status

error

eth_type

ethertype

F

failure_indices

field_count

field_name

fields

file_write_enabled

doca_telemetry_file_write_attr_t

flags

doca_job

doca_comm_channel_init_attr

doca_flow_actions

doca_flow_match

doca_flow_monitor

fwd

doca_flow_grpc_fwd

G

gre_key

doca_flow_tun

grpc_address

doca_argp_program_general_config

gtp_teid

doca_flow_tun

H

has_encap

doca_flow_actions

has_mkey

doca_regex_buffer

I

id

doca_regex_job_request

doca_regex_job_response

in_dst_ip

doca_flow_match

in_dst_mac

doca_flow_match

in_dst_port

doca_flow_match

in_eth_type

doca_flow_match

in_l4_type

doca_flow_match

in_src_ip

doca_flow_match

in_src_mac

doca_flow_match

in_src_port
in_tcp_flags
in_vlan_id
info
init_fn
inner
ipc_enabled
ipc_max_reconnect_time_msec
ipc_max_reconnect_tries
ipc_socket_timeout_msec
ipc_sockets_dir
ipv4
ipv4_addr
ipv6
ipv6_addr
is_cli_only
is_dpdk
is_grpc
is_mandatory
is_root

L

l4_dport

l4_protocol

l4_sport

length

log_level

long_flag

M

match

match_mask

match_start

matched

matches

max_file_age

max_file_size

max_job_size

max_packets_per_queue

max_sig_match_len

maxmsgs

mem_regs

N

nb_matches
   doca_dpi_stat_info
nb_meters
   doca_flow_resources
nb_other_l4
   doca_dpi_stat_info
nb_other_l7
   doca_dpi_stat_info
nb_queues
   doca_dpi_config_t
nb_scanned_pkts
   doca_dpi_stat_info
nb_ssl_parser_based
   doca_dpi_stat_info
nb_tcp_based
   doca_dpi_stat_info
nb_udp_based
   doca_dpi_stat_info
netflow_collector_addr
   doca_telemetry_netflow_send_attr_t
netflow_collector_port
   doca_telemetry_netflow_send_attr_t
next
   doca_regex_match
next_pipe
   doca_flow_fwd
next_pipe_id
   doca_flow_grpc_fwd
nr_shared_resources
   doca_flow_cfg
num_bytes_to_copy
   doca_dma_job_memcpy
num_failures
   doca_regex_compiled_rules_result
num_matches
   doca_regex_job_response
num_of_queues
   doca_flow_fwd
num_successful
   doca_regex_compiled_rules_result

O

offset
doca_flow_action_field

opaque_events_enabled
doca_telemetry_opaque_events_attr_t

out_dst_ip
doca_flow_match

out_dst_mac
doca_flow_match

out_dst_port
doca_flow_match

out_eth_type
doca_flow_match

out_l4_type
doca_flow_match

out_src_ip
doca_flow_match

out_src_mac
doca_flow_match

out_src_port
doca_flow_match

out_tcp_flags
doca_flow_match

out_vlan_id
doca_flow_match

outer
doca_flow_action_descs

P

pd
doca_regex_device

pipe_id
doca_flow_grpc_response
doca_flow_grpc_bindable_obj

pkt
doca_dpi_result
doca_dpi_grpc_result

pmi_supported
doca_regex_dev_caps

port
doca_flow_pipe_cfg

port_id
    doca_flow_grpc_bindable_obj
    doca_flow_port_cfg
    doca_flow_grpc_pipe_cfg
    doca_flow_fwd
priv_data_size
    doca_flow_port_cfg
program_compiled_rules_fn
    doca_regex_device
protocol
    doca_flow_tun

Q

query_dev_caps_fn
    doca_regex_device
query_dev_info_fn
    doca_regex_device
queue_depth
    doca_flow_cfg
queues
    doca_flow_cfg

R

resource
    doca_flow_cfg
result
    doca_event
    doca_dma_memcpy_result
rss_flags
    doca_flow_fwd
rss_mark
    doca_flow_fwd
rss_queues
    doca_flow_fwd
rule_group_ids
    doca_regex_job_request
rule_id
    doca_regex_match

S

seg_len
    doca_dpi_grpc_generic_packet

segment
    doca_dpi_grpc_generic_packet
short_flag
    doca_argp_param
sig_id
    doca_dpi_sig_info
    doca_dpi_sig_data
source_id
    doca_telemetry_source_name_attr_t
source_tag
    doca_telemetry_source_name_attr_t
src_buff
    doca_dma_job_memcpy
src_ip
    doca_dpi_parsing_info
    doca_flow_encap_action
    doca_flow_action_descs_packet
src_mac
    doca_flow_encap_action
    doca_flow_action_descs_packet
src_port
    doca_flow_action_descs_packet
status_flags
    doca_regex_job_response
    doca_dpi_grpc_result
    doca_dpi_result
success
    doca_flow_grpc_response

T

total_bytes
    doca_flow_query
total_pkts
    doca_flow_query
ttl
    doca_flow_action_descs_packet
tun
    doca_flow_match
    doca_flow_encap_action
tunnel
    doca_flow_action_descs
type
    doca_flow_pipe_cfg