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:

2.1. APSH

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_lib_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_lib_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_EXPERIMENTAL int doca_apsh_attestation_get (doca_apsh_process *process, const char *exec_hash_map_path, doca_apsh_attestationattestation)

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

Returns

Size of the array, error code on negative value.

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_EXPERIMENTAL int doca_apsh_attst_refresh (doca_apsh_attestationattestation)

refresh single attestation handler of a process with new snapshot

Parameters

attestation

single attestation handler to refresh

Returns

Size of the array, error code on negative value.

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_EXPERIMENTAL int doca_apsh_dma_dev_set (doca_apsh_ctx *ctx, const char *dma_dev_name)

Set apsh dma device.

Parameters

ctx
apsh handler

dma_dev_name
device name with the capabilities of dma

Returns

0 on success, error code otherwise.

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_EXPERIMENTAL int doca_apsh_libs_get (doca_apsh_process *process, doca_apsh_liblibs)

Get array of current process loadable libraries.

Parameters

process
Process handler

libs
Array of libs opaque pointers of the process

Returns

Size of the array, error code on negative value.

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_EXPERIMENTAL int doca_apsh_module_get (doca_apsh_system *system, doca_apsh_modulemodules)

Get array of current modules installed on the system.

Parameters

system

System handler

modules

Array of module opaque pointers of the systems

Returns

Size of the array, error code on negative value.

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 int doca_apsh_proc_refresh (doca_apsh_process *process)

refresh single process handler with new snapshot

Parameters

process
single process handler to refresh

Returns

0 on success, error code otherwise.

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 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_EXPERIMENTAL int doca_apsh_processes_get (doca_apsh_system *system, doca_apsh_processprocesses)

Get array of current processes running on the system.

Parameters

system
System handler
processes
Array of process opaque pointers of the systems

Returns

Size of the array, error code on negative value.

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 int 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

0 on success, error code otherwise.

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL int doca_apsh_start (doca_apsh_ctx *ctx)

Start apsh handler.

Parameters

ctx
App Shield handler

Returns

0 on success, error code otherwise.

Description

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

__DOCA_EXPERIMENTAL int 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 defrent PCI 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

0 on success, error code otherwise.

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL int 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

0 on success, error code otherwise.

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL int doca_apsh_sys_os_type_set (doca_apsh_system *system, doca_apsh_system_layer os_type)

Set system os type.

Parameters

system
system handler
os_type
system os type - windows/linux

Returns

0 on success, error code otherwise.

Description

This is a must setter

__DOCA_EXPERIMENTAL int doca_apsh_sys_pcidev_set (doca_apsh_system *system, int bdf)

Set system net device.

Parameters

system
system handler

bdf
pci function name connected to the system to run apsh on ex: "0000:00:01.0" as long format or "00:01.0" as short format

Returns

0 on success, error code otherwise.

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL int 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

0 on success, error code otherwise.

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_EXPERIMENTAL int doca_apsh_system_start (doca_apsh_system *system)

Start system handler.

Parameters

system
system handler

Returns

0 on success, error code otherwise.

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, pcidev. 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_EXPERIMENTAL int doca_apsh_threads_get (doca_apsh_process *process, doca_apsh_threadthreads)

Get array of current process threads.

Parameters

process

Process handler

threads

Array of threads opaque pointers of the process

Returns

Size of the array, error code on negative value.

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_EXPERIMENTAL int doca_apsh_vads_get (doca_apsh_process *process, doca_apsh_vadvads)

Get array of current process vads - virtual address descriptor.

Parameters

process
Process handler

vads
Array of vads opaque pointers of the process

Returns

Size of the array, error code on negative value.

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)__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)__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)__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)__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)__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. Compatibility Management

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")))

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

2.3. 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

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

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

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_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.

2.4. flow

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

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

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_OOM
Out of memory
DOCA_ERROR_PORT
Port error

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_flags

doca flow match flags

Values

DOCA_FLOW_MATCH_TCP_FIN
match tcp packets with Fin flag

enum doca_flow_port_type

doca flow port type

Values

DOCA_FLOW_PORT_DPDK_BY_ID
dpdk port by mapping id

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

__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_flow_pipe *doca_flow_create_control_pipe (doca_flow_port *port, doca_flow_error *error)

Create control pipe.

Parameters

port
Port struct.
error
Output error, set doca_flow_error for details.

Returns

pipe handler or NULL on failure.

Description

Control pipe is a special type of pipe that can have dynamic matches and forward with priority. Number of entries is limited (<64).

__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 void doca_flow_dump_pipe (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 and all entries information belong to this port.

__DOCA_EXPERIMENTAL void doca_flow_flush_pipe (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 int doca_flow_handle_aging (uint16_t queue, uint64_t quota, doca_flow_aged_query *entries, int len)

Handle aging of flows in queue.

Parameters

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, 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.
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, doca_flow_pipe_entry *entry)

Free one pipe entry.

Parameters

pipe_queue
Queue identifier.
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 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

2.5. flow net define

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_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.

2.6. Logging Management

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_log (uint32_t level, uint32_t source, 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.
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_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_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_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_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.

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_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_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_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

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, 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_REGISTER static int log_id; \ static void __attribute__((constructor(65535), 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.

2.7. NetFlow

DOCA lib for exporting a netflow packet to a netflow collector.

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 the conf file and invoking the init function, the lib's send function can be called with netflow struct to send a netflow packet with the format to the collector of choice, as specified in the conf file. The lib uses the netflow protocol specified by cisco.

See also:

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

Conf File structure:

doca_netflow.conf

[doca_netflow_conf]

target = <hostname = name/ipv4/ipv6>:<port = integer>

source_id = <ID = integer>

version = <version = 9>

doca_netflow_default.conf

[doca_netflow_conf]

target = 127.0.0.1:2055

source_id = 10

version = 9

Limitations:

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

struct doca_netflow_default_record

Flow record, represent a flow at specific moment, usually after a flow ends or after some timeout. Each one is a data record that will appear in the collector. This template is based on V5 fields with additional V9 fields.

struct doca_netflow_flowset_field

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

struct doca_netflow_template

Template for the records. struct record_example { uint32_t src_addr_V4; uint32_t dst_addr_V4; } struct doca_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_netflow_template template = { .field_count = 2; .fields = fields; };.

struct doca_netflow_default_record ::packed

Flow record, represent a flow at specific moment, usually after a flow ends or after some timeout. Each one is a data record that will appear in the collector. This template is based on V5 fields with additional V9 fields.

Note:

all fields are in network byte order.

__DOCA_EXPERIMENTAL void doca_netflow_exporter_destroy (void)

Free the exporter memory and close the connection.

__DOCA_EXPERIMENTAL int doca_netflow_exporter_init (const char *netflow_conf_file)

Init exporter memory, set configs and open connection.

Parameters

netflow_conf_file

Doca netflow configuration file pointer including a section marked as [doca_netflow_conf], if a NULL pointer is given will use the default path, as defined by DOCA_NETFLOW_CONF_DEFAULT_PATH. This function can be called again only after doca_netflow_exporter_destroy was called.

Returns

0 on success, error code otherwise.

__DOCA_EXPERIMENTAL int doca_netflow_exporter_send (const doca_netflow_template *netflow_template, const void **records, size_t length, int *error)

Sending netflow records. Need to init first.

Parameters

netflow_template

Template pointer for how the records are structured. for more info reffer to doca_netflow_template.

records

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

length

Records array size.

error

If return value is -1 populate this field with the error.

Returns

Number of records sent, -1 on error.

Description

Note:

- if the return value is positive but not equal to length then just some of the records were sent. The send function should run again with the remaining records. Please reffer to the example.
- When sending more then 30 records the lib splits the records to multiple packets because a single packet can only send up to 30 records (Netflow protocol limit)

doca_netflow_template *doca_netflow_template_default_get (void)

Return a default doca_netflow_template for use in send function, if using default template use doca_netflow_default_record struct for records.

Returns

pointer containing the default template

#define DOCA_NETFLOW_CONF_DEFAULT_PATH "/etc/doca_netflow.conf"

default conf path to look for

2.8. Telemetry Service Library

DOCA lib for exporting events to the telemetry service.

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_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_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.

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().

__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_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.

2.9. Version Management

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

const char *doca_version (void)

Function returning version string.

Returns

version string, using the format major.minor.patch

#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 2

Minor version number 0-255.

#define DOCA_VER_PATCH 6

Patch version number 0-999.

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

Return 1 if the version specified is equal to current.

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

Return 1 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.

Chapter 3. Data Structures

Here are the data structures with brief descriptions:

doca_dpi_config_t
DPI init configuration

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_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_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_tun
Doca flow tunnel information
doca_netflow_default_record
Flow record, represent a flow at specific moment, usually after a flow ends or after some timeout. Each one is a data record that will appear in the collector. This template is based on V5 fields with additional V9 fields
doca_netflow_flowset_field
One field in netflow template, please look at doca_netflow_types for type macros
doca_netflow_template
Template for the records. struct record_example { uint32_t src_addr_V4; uint32_t dst_addr_V4; } struct doca_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_netflow_template template = { .field_count = 2; .fields = fields; };
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_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_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

3.2. doca_dpi_parsing_info Struct Reference

L2-L4 flow information.

doca_dpi_parsing_info::l@0 doca_dpi_parsing_info::dst_ip

IP destination address

__be16 doca_dpi_parsing_info::ethertype

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

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

doca_dpi_parsing_info::@1 doca_dpi_parsing_info::src_ip

IP source address

3.3. 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

3.4. 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

3.5. 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

3.6. 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

3.7. 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

bool doca_flow_actions::has_encap

when true, will do encap

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

3.8. 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

3.9. doca_flow_cfg Struct Reference

doca flow global configuration

bool doca_flow_cfg::aging

when true, aging is handled by doca

bool doca_flow_cfg::is_hairpin

when true, the fwd will be hairpin queue

uint16_t doca_flow_cfg::queues

queue id for each offload thread

uint32_t doca_flow_cfg::total_sessions

total flows count

3.10. 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

3.11. 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

3.12. 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

3.13. 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

3.14. 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

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

doca_be16_t doca_flow_match::in_src_port

inner layer 4 source port if tunnel is used

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

struct doca_flow_tun doca_flow_match::tun

tunnel info

doca_be16_t doca_flow_match::vlan_id

outer vlan id

3.15. doca_flow_monitor Struct Reference

doca monitor action configuration

uint32_t doca_flow_monitor::aging

aging time in seconds.

uint64_t doca_flow_monitor::cir

Committed Information Rate (bytes/second).

uint8_t doca_flow_monitor::flags

indicate which actions be included

uint32_t doca_flow_monitor::id

meter id

uint64_t doca_flow_monitor::user_data

aging user data input.

3.16. doca_flow_pipe_cfg Struct Reference

pipeline configuration

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

doca_flow_port *doca_flow_pipe_cfg::port

port for the pipeline

3.17. 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

3.18. 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

3.19. 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

enumdoca_flow_tun_type doca_flow_tun::type

tunnel type

doca_be32_t doca_flow_tun::vxlan_tun_id

vxlan vni(24) + reserved (8).

3.20. doca_netflow_default_record Struct Reference

Flow record, represent a flow at specific moment, usually after a flow ends or after some timeout. Each one is a data record that will appear in the collector. This template is based on V5 fields with additional V9 fields.

Note:

all fields are in network byte order.

char doca_netflow_default_record::application_name

Name associated with a classification

__be32 doca_netflow_default_record::d_octets

Octets sent in Duration.

__be32 doca_netflow_default_record::d_pkts

Packets sent in Duration

__be32 doca_netflow_default_record::dst_addr_v4

Destination IPV4 Address

in6_addr doca_netflow_default_record::dst_addr_v6

Destination IPV6 Address

__be16 doca_netflow_default_record::dst_as

originating AS of destination address

uint8_t doca_netflow_default_record::dst_mask

destination address prefix mask bits

__be16 doca_netflow_default_record::dst_port

TCP/UDP destination port number or equivalent

__be32 doca_netflow_default_record::first

SysUptime at start of flow

__be64 doca_netflow_default_record::flow_id

This identifies a transaction within a connection

__be16 doca_netflow_default_record::input

Input interface index

__be32 doca_netflow_default_record::last

and of last packet of flow

__be32 doca_netflow_default_record::next_hop_v4

Next hop router's IPV4 Address

in6_addr doca_netflow_default_record::next_hop_v6

Next hop router's IPV6 Address

__be16 doca_netflow_default_record::output

Output interface index

uint8_t doca_netflow_default_record::protocol

IP protocol type (for example, TCP = 6;UDP = 17)

__be32 doca_netflow_default_record::src_addr_v4

Source IPV4 Address

in6_addr doca_netflow_default_record::src_addr_v6

Source IPV6 Address

__be16 doca_netflow_default_record::src_as

originating AS of source address

uint8_t doca_netflow_default_record::src_mask

source address prefix mask bits

__be16 doca_netflow_default_record::src_port

TCP/UDP source port number or equivalent

uint8_t doca_netflow_default_record::tcp_flags

Cumulative OR of tcp flags

uint8_t doca_netflow_default_record::tos

IP Type-of-Service

3.21. doca_netflow_flowset_field Struct Reference

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

int doca_netflow_flowset_field::length

field len in bytes (see link) - will be converted to uint16

int doca_netflow_flowset_field::type

field number id (see link) - will be converted to uint16

3.22. doca_netflow_template Struct Reference

Template for the records. struct record_example { uint32_t src_addr_V4; uint32_t dst_addr_V4; } struct doca_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_netflow_template template = { .field_count = 2; .fields = fields; };.

Note:

all fields are in network byte order.

int doca_netflow_template::field_count

number of fields in 'fields' array - will be converted to uint16

doca_netflow_flowset_field *doca_netflow_template::fields

array of field info

3.23. 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 for where the data and schema will be stored.

3.24. 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).

3.25. 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.

3.26. 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.

3.27. 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.

3.28. 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.

3.29. 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

actions
- doca_flow_pipe_cfg

aging
- doca_flow_monitor
- doca_flow_cfg

application_name
- doca_netflow_default_record

array_length
- doca_telemetry_field_info_t

B

buffer_size
- doca_telemetry_buffer_attr_t

C

cir
- doca_flow_monitor

D

d_octets
- doca_netflow_default_record

d_pkts
- doca_netflow_default_record

data_root
- doca_telemetry_buffer_attr_t

dec_ttl
- doca_flow_actions

E

F

ipv4
doca_dpi_parsing_info
ipv4_addr
doca_flow_ip_addr
ipv6
doca_dpi_parsing_info
ipv6_addr
doca_flow_ip_addr
is_hairpin
doca_flow_cfg
is_root
doca_flow_pipe_cfg

L

l4_dport
doca_dpi_parsing_info
l4_protocol
doca_dpi_parsing_info
l4_sport
doca_dpi_parsing_info
last
doca_netflow_default_record
length
doca_netflow_flowset_field

M

match
doca_flow_pipe_cfg
match_mask
doca_flow_pipe_cfg
matched
doca_dpi_result
max_file_age
doca_telemetry_file_write_attr_t
max_file_size
doca_telemetry_file_write_attr_t
max_packets_per_queue
doca_dpi_config_t
max_sig_match_len
doca_dpi_config_t
message
doca_flow_error

N

num_of_queues
doca_flow_fwd

O

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
output
doca_netflow_default_record

P

pkt
doca_dpi_result
port
doca_flow_pipe_cfg
port_id
doca_flow_fwd
doca_flow_port_cfg
priv_data_size
doca_flow_port_cfg
protocol
doca_netflow_default_record

Q

queues
doca_flow_cfg

R

rss_flags
- doca_flow_fwd

rss_mark
- doca_flow_fwd

rss_queues
- doca_flow_fwd

S

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_addr_v4
- doca_netflow_default_record

src_addr_v6
- doca_netflow_default_record

src_as
- doca_netflow_default_record

src_ip
- doca_dpi_parsing_info
- doca_flow_encap_action

src_mac
- doca_flow_encap_action

src_mask
- doca_netflow_default_record

src_port
- doca_netflow_default_record

status_flags
- doca_dpi_result

T

tcp_flags
- doca_netflow_default_record

tos
- doca_netflow_default_record

total_bytes
- doca_flow_query

total_pkts
- doca_flow_query

total_sessions
- doca_flow_cfg

tun
- doca_flow_match
- doca_flow_encap_action

type
- doca_flow_ip_addr
- doca_flow_fwd
- doca_netflow_flowset_field
- doca_flow_tun
- doca_flow_error
- doca_flow_port_cfg

type_name
- doca_telemetry_field_info_t

U

user_data
- doca_dpi_result
- doca_flow_aged_query
- doca_flow_monitor

V

vlan_id
- doca_flow_match

vxlan_tun_id
- doca_flow_tun

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

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