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:

- Compatibility Management
- Deep packet inspection
- NetFlow
- Flow
- Logging Management
- Version Management

2.1. 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.

2.2. Deep packet inspection

Deep packet inspection (DPI) is a method of examining the full content of data packets as they traverse a monitored network checkpoint.

DPI provides a more robust mechanism for enforcing network packet filtering as it can be used to identify and block a range of complex threats hiding in network data streams more

accurately. This includes: • Malicious applications • Malware data exfiltration attempts • Content policy violations • Application recognition • Load balancing

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 enqueue the flows.

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. Typically, the first packet will arrive from the initiator.
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 form 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, the DPI engine will increase ref count in the mbuf. User must not change or reuse 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.

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 DPI queue on which the flow was created.

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

0 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.

2.3. NetFlow

Doca lib for export 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 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

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 end 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_exmaple { 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 end 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 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 configure file pointer including a section marked as [doca_netflow_conf], if a NULL pointer is given then look at default path in 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 *template, const void **records, size_t length, int *error)

Sending netflow records. Need to init first.

Parameters

template

Template pointer how the records are structed. 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 have sent. the send function sould run again with the remaining records. please reffer to the exmaple.
- When sending more then 30 records the lib split the records to multiple packets because packet can 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

2.4. Flow

DOCA FLOW is the most fundamental API for building generic execution pipes in HW. The library provides an API for building a set of pipes, where each pipe consists of match criteria, monitoring, and a set of actions. Pipes can be connected where after pipe-defined actions are executed, the packet may proceed to another pipe.

struct doca_flow_actions

doca flow actions information

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_PIPE_BUILD_ITEM
Build pipe match items error
DOCA_ERROR_PIPE_BUILD_ACTION
Build pipe actions 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 *doca_flow_create_pipe (const doca_flow_pipe_cfg *cfg, const doca_flow_fwd *fwd, doca_flow_error *error)

Create one new pipe.

Parameters

cfg

Pipe configuration

fwd

Fwd configuration for the pipe

error

Output error

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 those components:

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

this API will create the pipe, but not actual start the HW offload.

__DOCA_EXPERIMENTAL void doca_flow_destroy (void)

Destroy the doca flow.

Description

Release all the resource used by doca flow .

It must be invoked at the end of application exit.

__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 matched 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 resource 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 out put 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 belong to the port.

__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 initialize function for doca flow, will initialize all resource used by doca flow.

It must be invoked first before any function in the API. One time call, used for doca flow initialize 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
indentify each queue

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

Returns

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

Description

when one packet match to one pipe, will start HW offload, pipe only define which files to match, when do offload, we need detail information from packets, or we need set some specific actions that pipe not define, the parameters include:

match: the detail packets fields according to the pipe definition. actions: the real actions according to the pipe definition. monitor: define the monitor actions if pipe not define it. fwd: define the forward action if pipe not define it.

This API will do the actual HW offload, with the input detail packets information.

__DOCA_EXPERIMENTAL int doca_flow_pipe_rm_entry (uint16_t pipe_queue, doca_flow_pipe_entry *entry)

Free one pipe entry.

Parameters

pipe_queue
indentify each queue

entry
the pipe entry be removed

Returns

0 on success, negative on fail.

Description

This API will free the pipe entry, cancel HW offload. Application will hold the entry pointer when create one entry, if no need use this offload, for example, the entry aged, then 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 configuration, will create one port in doca flow layer, allocate all resources used by this port, and create the default offload flows include 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 fail.

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 be queried

query_stats
data retrieved by the query

Returns

0 on success, negative on fail.

Description

Query the packet statistics about specific pipe entry

2.5. 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

doca_log (uint32_t level, uint32_t type, const char *format, ...)

Parameters

level
Log level enum DOCA_LOG_LEVEL

type
The log type identifier defined by doca_log_type_register

format
printf(3) arguments, format and variables

Description

Generates a log message.

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

doca_log_global_level_set (uint32_t level)

Set the global log level.

Parameters

level
Log level enum DOCA_LOG_LEVEL

Description

Dynamically change global log level, any log under this type will be showen

doca_log_stream_redirect (FILE *stream)

redirect the logger to different stream

Parameters

stream
Pointer to the stream. can't be NULL.

Returns

0 on success, error code otherwise.

Description

Dynamically change the logger stream. can be file pointer or any stream. The default stream is stderr.

uint32_t doca_log_type_register (const char *type_name)

Parameters

type_name

The string identifying the log type. should be in an hirechy form (i.e. DPI::Parser)

Returns

The log type identifier. negative for err.

Description

Register a log type

Will return the number associate with the log type. Log type will be showen in the logs.

#define DOCA_DLOG_CRIT DOCA_DLOG(CRIT, __VA_ARGS__)

Generates an CRIT development log message.

Will generate critical log for development porpeses. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation veribales. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, will be removed.

#define DOCA_DLOG_DBG DOCA_DLOG(DEBUG, __VA_ARGS__)

Generates an DEBUG development log message.

Will generate debug log for development porpeses. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation veribales. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, will be removed.

#define DOCA_DLOG_ERR DOCA_DLOG(ERROR, __VA_ARGS__)

Generates an ERROR development log message.

Will generate error log for development porpeses. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation veribales. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, will be removed.

#define DOCA_DLOG_INFO DOCA_DLOG(INFO, __VA_ARGS__)

Generates an INFO development log message.

Will generate info log for development porpeses. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation veribales. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, will be removed.

#define DOCA_DLOG_WARN DOCA_DLOG(WARNING, __VA_ARGS__)

Generates an WARNING development log message.

Will generate warning log for development porpeses. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation veribales. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, will be removed.

#define DOCA_LOG do { \ doca_log(DOCA_LOG_LEVEL_##level, log_id, __VA_ARGS__); \ } while (0)

Generates a log message.

The DOCA_LOG() is the main log function for logging, This will effect performace. Consider using DOCA_DLOG for the option to remove it on the final compilation. Consider using the specific level DOCA_LOG for better code readebility (i.e. DOCA_LOG_ERROR)

#define DOCA_LOG_CRIT DOCA_LOG(CRIT, __VA_ARGS__)

Generates an CRITICAL log message.

Will generate critical log, This will effect performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_DBG DOCA_LOG(DEBUG, __VA_ARGS__)

Generates an DEBUG log message.

Will generate debug log, This will effect performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_ERR DOCA_LOG(ERROR, __VA_ARGS__)

Generates an ERROR log message.

Will generate error log, This will effect performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_INFO DOCA_LOG(INFO, __VA_ARGS__)

Generates an INFO log message.

Will generate info log, This will effect performace. 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_type_register(#TYPE); \ }

Registers log type on program start.

Should be used to register the log type. For example

DOCA_LOG_REGISTER(dpi)

void foo { DOCA_LOG_INFO("Message"); }

#define DOCA_LOG_WARN DOCA_LOG(WARNING, __VA_ARGS__)

Generates an WARNING log message.

Will generate warning log, This will effect performace. Consider using DOCA_DLOG for the option to remove it on the final compilation.

2.6. Version Management

Define function to get doca version and version compare.

const char *doca_version (void)

Function returning version string.

Returns

string version number of a 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 0

Major version number 0-255.

#define DOCA_VER_MINOR 1

Minor version number 0-255.

#define DOCA_VER_PATCH 0

Patch version number 0-255.

#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 ((major) << 16 | (minor) << 8 | (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_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 end 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_exmaple { 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; };

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.

__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

in6_addr doca_dpi_parsing_info::ipv6

Ipv6 destination 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

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 l5 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 tco 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_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

3.9. 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

3.10. 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

3.11. 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

3.12. 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

3.13. 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

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

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

3.14. 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

3.15. doca_flow_pipe_cfg Struct Reference

pipeline configuration

doca_flow_actions *doca_flow_pipe_cfg::actions

actions for the pipeline

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

3.16. 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

3.17. 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

3.18. doca_flow_tun Struct Reference

doca flow tunnel information

doca_be32_t doca_flow_tun::gre_key

gre key

enum doca_flow_tun_type doca_flow_tun::type

tunnel type

uint32_t doca_flow_tun::vxlan_tun_id

vxlan vni

3.19. doca_netflow_default_record Struct Reference

Flow record, represent a flow at specific moment, usually after a flow end 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

3.20. 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

3.21. doca_netflow_template Struct Reference

Template for the records. struct record_exmaple { 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

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

C

cir
doca_flow_monitor

D

d_octets
doca_netflow_default_record
d_pkts
doca_netflow_default_record
dec_ttl
doca_flow_actions
decap
doca_flow_actions
devargs
doca_flow_port_cfg
dst_addr_v4
doca_netflow_default_record
dst_addr_v6
doca_netflow_default_record

E

F

G

H

I

in_dst_port
doca_flow_match
in_l4_type
doca_flow_match
in_src_ip
doca_flow_match
in_src_port
doca_flow_match
info
doca_dpi_result
input
doca_netflow_default_record
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

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

N

next_hop_v4
doca_netflow_default_record
next_hop_v6
doca_netflow_default_record
next_pipe
doca_flow_fwd
num_of_queues
doca_flow_fwd

O

out_dst_ip
doca_flow_match
out_dst_mac
doca_flow_match
out_dst_port
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

src_addr_v4
- doca_netflow_default_record

src_addr_v6
- doca_netflow_default_record

src_as
- doca_netflow_default_record

src_ip
- 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

U

V

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. 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