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. App Shield

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

const __DOCA_EXPERIMENTAL void *__doca_apsh_attst_info_get (doca_apsh_attestation *attestation, 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, recommended to use doca_apsh_attestation_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_envar_info_get (doca_apsh_envar *envar, doca_apsh_envar_attr attr)

Shadow function - get attribute value for an environment variable.

Parameters

envar
single envar handler

attr
Attribute to get the info on the envar

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_envar_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_handle_info_get (doca_apsh_handle *handle, doca_apsh_handle_attr attr)

Shadow function - get attribute value for a handle.

Parameters

handle
single handle handler

attr
Attribute to get the info on the handle

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_handle_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_ldrmodule_info_get (doca_apsh_ldrmodule *ldrmodule, doca_apsh_ldrmodule_attr attr)

Shadow function - get attribute value for a modules.

Parameters

ldrmodule
single ldrmodule handler

attr
Attribute to get the info on the module

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_ldrmodule_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_lib_info_get (doca_apsh_lib *lib, 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, recommended to use doca_apsh_lib_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_module_info_get (doca_apsh_module *module, 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, recommended to use doca_apsh_mod_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_netscan_info_get (doca_apsh_netscan *connection, doca_apsh_netscan_attr attr)

Shadow function - get attribute value for a connection.

Parameters

connection
single connection handler

attr
Attribute to get the info on the connection

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_netscan_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_privilege_info_get (doca_apsh_privilege *privilege, doca_apsh_privilege_attr attr)

Shadow function - get attribute value for a privilege.

Parameters

privilege
single privilege handler

attr
Attribute to get the info on the privilege

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_privilege_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_process_info_get (doca_apsh_process *process, 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, recommended to use doca_apsh_process_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_process_parameters_info_get (doca_apsh_process_parameters *process_parameters, doca_apsh_process_parameters_attr attr)

Shadow function - get attribute value for a process-parameter.

Parameters

process_parameters
single process_parameters handler

attr
Attribute to get the info on the process_parameters

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_process_parameters_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_sid_info_get (doca_apsh_sid *sid, doca_apsh_sid_attr attr)

Shadow function - get attribute value for a SID.

Parameters

sid
single SID handler

attr
Attribute to get the info on the SID

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_sid_info_get

doca_error_t __doca_apsh_sys_config (doca_apsh_system *system, doca_apsh_system_config_attr attr, void *value)

Shadow function - configure attribute value for a system.

Parameters

system
system handler

attr
Attribute to set in the system

value
the value to set

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_SUPPORTED - if attr was OS type and an unsupported OS type had been received.
- DOCA_ERROR_NO_MEMORY - if memory allocation failed.
- DOCA_ERROR_BAD_STATE - if system is already started.

Description

Do not use this function, recommended to use doca_apsh_sys_config

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

Shadow function - get attribute value for a thread.

Parameters

thread
single thread handler

attr
Attribute to get the info on the thread

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_thread_info_get

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

Shadow function - get attribute value for a vad.

Parameters

vad
single vad handler

attr
Attribute to get the info on the vad

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_vad_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_yara_info_get (doca_apsh_yara *yara, doca_apsh_yara_attr attr)

Shadow function - get attribute value for a yara.

Parameters

yara
single yara handler

attr
Attribute to get the info on the yara

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use doca_apsh_yara_info_get

__DOCA_EXPERIMENTAL void doca_apsh_attestation_free (doca_apsh_attestation **attestation)

Destroys a attestation context.

Parameters

attestation
Attestation opaque pointer of the process to destroy

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

Get current process attestation.

Parameters

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if modules list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.

- DOCA_ERROR_NOT_FOUND - if process hasn't been found.

Description

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

doca_error_t doca_apsh_attst_refresh (doca_apsh_attestationattestation, int *attestation_size)

refresh single attestation handler of a process with new snapshot

Parameters

attestation
single attestation handler to refresh

attestation_size
Output param, will contain size of attestation array on success.

Returns

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

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

Description

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

__DOCA_EXPERIMENTAL doca_apsh_ctx *doca_apsh_create (void)

Create a new apsh handler.

Returns

apsh context required for creating system handler, NULL on failure

Description

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

__DOCA_EXPERIMENTAL void doca_apsh_destroy (doca_apsh_ctx *ctx)

Free the APSH memory and close connections.

Parameters

ctx
apsh context to destroy

doca_error_t doca_apsh_dma_dev_set (doca_apsh_ctx *ctx, doca_dev *dma_dev)

Set apsh dma device.

Parameters

ctx
apsh handler
dma_dev
doca device with dma capabilities, please refer to doca_dev.h

Returns

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

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

Description

This is a Mandatory setter

__DOCA_EXPERIMENTAL void doca_apsh_envars_free (doca_apsh_envar **envars)

Destroys a envars context.

Parameters

envars
Array of envars opaque pointers of the process to destroy

doca_error_t doca_apsh_envars_get (doca_apsh_process *process, doca_apsh_envarenvars, int *envars_size)

Get array of current process environment variables.

Parameters

process
Process handler

envars
Array of environment variables opaque pointers of the process. in case process doesn't have any envars, will return NULL.

envars_size
Output param, will contain size of envars array on success.

Returns

DOCA_SUCCESS - in case of success (including the case envars_size is zero). doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if envars list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to envars array.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

Description

This function is multi-threaded compatible with different system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, the function allocates this array, use doca_apsh_envars_free to free it.

Note:

currently supported only for windows systems.

__DOCA_EXPERIMENTAL void doca_apsh_handles_free (doca_apsh_handle **handles)

Destroys a handles context.

Parameters

handles
Array of handles opaque pointers of the process to destroy

doca_error_t doca_apsh_handles_get (doca_apsh_process *process, doca_apsh_handlehandles, int *handles_size)

Get array of current process handles.

Parameters

process
Process handler
handles
Array of handles opaque pointers of the process. in case process doesn't have any handles, will return NULL.
handles_size
Output param, will contain size of handles array on success.

Returns

DOCA_SUCCESS - in case of success (including the case handles_size is zero). doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if handles list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to handles array.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

Description

This function is multi-threaded compatible with different 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.


Note:

currently supported only for windows systems.

__DOCA_EXPERIMENTAL void doca_apsh_ldrmodules_free (doca_apsh_ldrmodule **ldrmodules)

Destroys a ldrmodules context.

Parameters

ldrmodules
Array of ldrmodules opaque pointers of the process to destroy

doca_error_t doca_apsh_ldrmodules_get (doca_apsh_process *process, doca_apsh_ldrmoduleldrmodules, int *ldrmodules_size)

Get array of current process modules.

Parameters

process
Process handler

ldrmodules
Array of ldrmodules opaque pointers of the process. in case process doesn't have any modules, will return NULL.

ldrmodules_size
Output param, will contain size of ldrmodules array on success.

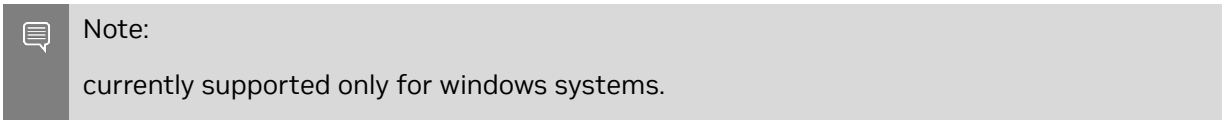
Returns

DOCA_SUCCESS - in case of success (including the case ldrmodules_size is zero).
doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if ldrmodules list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to ldrmodules array.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

Description

This function is multi-threaded compatible with different 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_libs_free (doca_apsh_lib **libs)

Destroys a libs context.

Parameters

libs
Array of libs opaque pointers of the process to destroy

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

Get array of current process loadable libraries.

Parameters

process
Process handler

libs
Array of libs opaque pointers of the process. in case process doesn't point to any libs, will return NULL.

libs_size
Output param, will contain size of libs array on success.

Returns

DOCA_SUCCESS - in case of success (including the case libs_size is zero). doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if libs list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to libs array.

Description

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

__DOCA_EXPERIMENTAL void doca_apsh_module_free (doca_apsh_module **modules)

Destroys a modules array.

Parameters

modules
Array of module opaque pointers of the systems to destroy

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

Get array of current modules installed on the system.

Parameters

system
System handler
modules
Array of module opaque pointers of the systems
modules_size
Output param, will contain size of modules array on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if modules list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.

Description

This function is multithreaded compatible with different 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_netscan_free (doca_apsh_netscan **connections)

Destroys a netscan context.

Parameters

connections
Array of connections opaque pointers of the system to destroy

doca_error_t doca_apsh_netscan_get (doca_apsh_system *system, doca_apsh_netscanconnections, int *connections_size)

Get array of current connections.

Parameters

system
System handler
connections
Pointer to array of connections opaque pointers of the system
connections_size
Output param, will contain size of connections array on success

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if connections list initialization failed or no regex device was set.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to connections array.
- DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type has been received (or unsupported OS build). list of supported builds: Windows 10 10240 x86 Windows 10

10586 x86 Windows 10 14393 x86 Windows 10 15063 x64 Windows 10 15063 x86 Windows 10 16299 x64 Windows 10 17134 x64 Windows 10 17134 x86 Windows 10 17763 x64 Windows 10 18362 x64 Windows 10 18363 x64 Windows 10 19041 x64 Windows 10 19041 x86

- DOCA_ERROR_BAD_STATE - if system isn't started yet.

Description

This function is multi-threaded compatible with different 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.

Note:

- currently supported only for systems with windows 10 build (such as: windows 10 and windows server 2019).
- this function requires the usage of regex device. (set it using doca_apsh_regex_dev_set)

__DOCA_EXPERIMENTAL void doca_apsh_privileges_free (doca_apsh_privilege **privileges)

Destroys a privileges context.

Parameters

privileges

Array of privileges opaque pointers of the process to destroy

doca_error_t doca_apsh_privileges_get (doca_apsh_process *process, doca_apsh_privilegeprivileges, int *privileges_size)

Get array of current process privileges.

Parameters

process

Process handler

privileges

Array of privileges opaque pointers of the process. in case process doesn't have any privileges, will return NULL.

privileges_size
Output param, will contain size of privileges array on success.

Returns

DOCA_SUCCESS - in case of success (including the case privileges_size is zero).
doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if privileges list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to privileges array.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

Description

This function is multi-threaded compatible with different 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.

Note:

currently supported only for windows systems.

__DOCA_EXPERIMENTAL void doca_apsh_process_parameters_free (doca_apsh_process_parameters *process_parameters)

Destroys a process-parameters context.

Parameters

process_parameters
process-parameters opaque pointer of the process

doca_error_t doca_apsh_process_parameters_get (doca_apsh_process *process, doca_apsh_process_parameters **process_parameters)

Get current process parameters.

Parameters

process

Process handler

process_parameters

Pointer of process-parameters opaque pointer of the process. In case process-parameters data are paged out, will return NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if process-parameters object initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot allocate memory to process-parameters object.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.
- DOCA_ERROR_BAD_STATE - in case the relevant memory is not present in the system memory.

Description

This function is multi-threaded compatible with different system context, meaning do not call this function simultaneously with the same system context. The return object is a snapshot, not a dynamic object, need to free it.

Note:

currently supported only for windows systems.

__DOCA_EXPERIMENTAL void doca_apsh_processes_free (doca_apsh_process **processes)

Destroys a process context.

Parameters

processes
Array of process opaque pointers of the systems to destroy

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

Get array of current processes running on the system.

Parameters

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if processes list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to processes array.

Description

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

doca_error_t doca_apsh_regex_dev_set (doca_apsh_ctx *ctx, doca_dev *regex_dev)

Set apsh regex device.

Parameters

ctx
apsh handler

regex_dev
doca device with the capabilities of regex

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This is not a Mandatory setter

Note:

currently this device is used only for windows systems.

__DOCA_EXPERIMENTAL void doca_apsh_sids_free (doca_apsh_sid **sids)

Destroys a SIDs context.

Parameters

sids
Array of SIDs opaque pointers of the process to destroy

doca_error_t doca_apsh_sids_get (doca_apsh_process *process, doca_apsh_sidsids, int *sids_size)

Get array of current process SIDs.

Parameters

process
Process handler

sids
Array of SIDs opaque pointers of the process. in case process doesn't have any SIDs, will return NULL.

sids_size
Output param, will contain size of SIDs array on success.

Returns

DOCA_SUCCESS - in case of success (including the case handles_size is zero). doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if SIDs list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to SIDs array.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

Description

This function is multi-threaded compatible with different 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.

Note:

currently supported only for windows systems.

doca_error_t doca_apsh_start (doca_apsh_ctx *ctx)

Start apsh handler.

Parameters

ctx
App Shield handler

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

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

doca_error_t doca_apsh_sys_dev_set (doca_apsh_system *system, doca_dev_rep *dev)

Set system device.

Parameters

system
system handler

dev
the device that is connected to the system to be queried. for example a vf that is connected to a vm or pf that is connected to the bare-metal. doca representor device from dma device configured in doca_apsh_dma_dev_set. to query the right device please refer to doca_dev.h for full options.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if system was already started.

Description

This is a Mandatory setter

doca_error_t doca_apsh_sys_kpgd_file_set (doca_apsh_system *system, const char *system_kpgd_file_path)

Set system kpgd file.

Parameters

system
system handler

system_kpgd_file_path
the path to kpgd file

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type had been received.

- DOCA_ERROR_BAD_STATE - if system was already started.

Description

This is not a must setter

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

Set system allowed memory regions.

Parameters

system
system handler

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.
- DOCA_ERROR_BAD_STATE - if system was already started.

Description

This is a Mandatory setter

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

Set system os symbol map.

Parameters

system
system handler

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.
- DOCA_ERROR_BAD_STATE - if system was already started.

Description

This is a Mandatory setter

doca_error_t doca_apsh_sys_os_type_set (doca_apsh_system *system, doca_apsh_system_os os_type)

Set system os type.

Parameters

system
system handler

os_type
system os type - windows/linux

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type had been received.
- DOCA_ERROR_BAD_STATE - if system was already started.

Description

This is a must setter

doca_error_t doca_apsh_sys_set_scan_window_size (doca_apsh_system *system, uint32_t scan_window_size)

Set system yara scan window size.

Parameters

system
system handler

scan_window_size
yara scan window size (in bytes) a condition on scan window size is: (window_scan_size % PAGE_SIZE == 0) or (PAGE_SIZE % window_scan_size == 0)

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This is not a must setter. Default size is 4KB.

doca_error_t doca_apsh_sys_set_scan_window_step (doca_apsh_system *system, uint32_t scan_window_step)

Set system yara scan window step.

Parameters

system
system handler

scan_window_step
yara scan window step (in bytes) a condition on scan window step is: window_scan_size % scan_window_step == 0

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This is not a must setter. Default size is 4KB. Since this setter is dependant on scan_window_size, make sure to call it afrer "doca_apsh_sys_set_scan_window_size".

__DOCA_EXPERIMENTAL doca_apsh_system *doca_apsh_system_create (doca_apsh_ctx *ctx)

Create a new system handler.

Parameters

ctx
apsh handler

Returns

returns system pointer, NULL on failure

Description

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

__DOCA_EXPERIMENTAL void doca_apsh_system_destroy (doca_apsh_system *system)

Destroy system handler.

Parameters

system
system context to destroy

Description

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

doca_error_t doca_apsh_system_start (doca_apsh_system *system)

Start system handler.

Parameters

system
system handler

Returns

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

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

Description

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

__DOCA_EXPERIMENTAL void doca_apsh_threads_free (doca_apsh_thread **threads)

Destroys a threads context.

Parameters

threads
Array of threads opaque pointers of the process to destroy

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

Get array of current process threads.

Parameters

process
Process handler
threads
Array of threads opaque pointers of the process. in case process doesn't have any threads, will return NULL.
threads_size
Output param, will contain size of threads array on success.

Returns

DOCA_SUCCESS - in case of success (including the case threads_size is zero). doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if threads list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to threads array.

Description

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

__DOCA_EXPERIMENTAL void doca_apsh_vads_free (doca_apsh_vad **vads)

Destroys a vads context.

Parameters

vads
Array of vads opaque pointers of the process to destroy

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

Get array of current process vads - virtual address descriptor.

Parameters

process
Process handler
vads
Array of vads opaque pointers of the process. in case process doesn't point to any vads, will return NULL.
vads_size
Output param, will contain size of vads array on success.

Returns

DOCA_SUCCESS - in case of success (including the case vads_size is zero). doca_error code - in case of failure:

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

Description

This function is multithreaded compatible with different 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_yara_free (doca_apsh_yara **yara_matches)

Destroys a yara context.

Parameters

yara_matches
Array of yara matches opaque pointers to destroy

doca_error_t doca_apsh_yara_get (doca_apsh_process *process, doca_apsh_yara_rule * *yara_rules_arr, uint32_t yara_rules_arr_size, uint64_t scan_type, doca_apsh_yarayara_matches, int *yara_matches_size)

Scan current process with yara rules. The scanning is done with a window size and step that are set by doca_apsh_sys_set_scan_window_size and doca_apsh_sys_set_scan_window_step.

Parameters

process
Process handler
yara_rules_arr
Array of type doca_apsh_yara_rule containing the rules to check against the process's memory
yara_rules_arr_size
Length of yara_rules_arr
scan_type
YARA scan type bitmask - to scan the whole vad tree or just heaps This will affect performance, please see enum doca_apsh_yara_scan_type
yara_matches
Point to array of yara matches opaque pointers. In case no yara matches were found, will return NULL.
yara_matches_size
Output param, will contain size of YARA array on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if yara matches list initialization failed.

- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to yara matches array.
- DOCA_ERROR_NOT_FOUND - if process hasn't been found.
- DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os or DPU.

Description

This function is multithreaded compatible with different 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.

Note:

1. Currently supported only for windows systems 2. Currently supported only on DPU with Ubuntu 22.04.

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

Get attribute value for a attestation.

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

#define doca_apsh_envar_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_envar_info_get(envar, attr))

Get attribute value for an environment variable.

Get the requested info from envar 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_handle_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_handle_info_get(handle, attr))

Get attribute value for a handle.

Get the requested info from handle 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_ldrmodule_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_ldrmodule_info_get(ldrmodule, attr))

Get attribute value for a ldrmodule.

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

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

Get attribute value for a lib.

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

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

Get attribute value for a module.

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

#define doca_apsh_netscan_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_netscan_info_get(connection, attr))

Get attribute value for a connection.

Get the requested info from connection 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_privilege_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_privilege_info_get(privilege, attr))

Get attribute value for a privilege.

Get the requested info from privilege 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_process_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_process_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_process_parameters_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_process_parameters_info_get(proce attr))

get attribute value for a process-parameter

Get the requested info from process_parameters 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_sid_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_sid_info_get(sid, attr))

Get attribute value for a SID.

Get the requested info from SID 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_sys_config (__doca_apsh_sys_config(system, attr, (void *) ((uintptr_t)value)))

configure attribute value for a system, such as: hashtest limit, symbols map ...

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

Get attribute value for a thread.

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

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

Get attribute value for a vad.

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

#define doca_apsh_yara_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_yara_info_get(yara, attr))

Get attribute value for a yara.

Get the requested info from yara 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. App Shield Attributes

DOCA App Shield attributes to query with get functions, see doca_apsh.h

enum doca_apsh_attestation_attr

doca app shield attestation attributes

Values

DOCA_APSH_ATTESTATION_PID
attestation process id
DOCA_APSH_ATTESTATION_COMM
attestation process name
DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA
attestation path of memory area
DOCA_APSH_ATTESTATION_PROTECTION
attestation protection
DOCA_APSH_ATTESTATION_START_ADDRESS
attestation start address
DOCA_APSH_ATTESTATION_END_ADDRESS
attestation end address
DOCA_APSH_ATTESTATION_PAGES_NUMBER
attestation process pages count in binary file
DOCA_APSH_ATTESTATION_PAGES_PRESENT
attestation pages present in memory
DOCA_APSH_ATTESTATION_MATCHING_HASHES
attestation pages hash match count from pages in memory
DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT
attestation hash data is present

enum doca_apsh_envar_attr

doca app shield envars attributes

Values

DOCA_APSH_ENVARS_PID
envars pid
DOCA_APSH_ENVARS_COMM
envars process name
DOCA_APSH_ENVARS_VARIABLE
envars variable
DOCA_APSH_ENVARS_VALUE
envars value
DOCA_APSH_ENVARS_WINDOWS_BLOCK = 1000
envars windows environment block address

enum doca_apsh_handle_attr

doca app shield handle attributes

Values

DOCA_APSH_HANDLE_PID
handle process id
DOCA_APSH_HANDLE_COMM
handle process name
DOCA_APSH_HANDLE_VALUE
handle value
DOCA_APSH_HANDLE_TABLE_ENTRY
handle table entry
DOCA_APSH_HANDLE_TYPE
handle type
DOCA_APSH_HANDLE_ACCESS
handle access
DOCA_APSH_HANDLE_NAME
handle name

enum doca_apsh_ldrmodule_attr

doca app shield LDR-Modules attributes

Values

DOCA_APSH_LDRMODULE_PID
ldrmodule process pid
DOCA_APSH_LDRMODULE_COMM
ldrmodule process name
DOCA_APSH_LDRMODULE_BASE_ADDRESS
ldrmodule base address
DOCA_APSH_LDRMODULE_LIBRARY_PATH
ldrmodule loaded library path
DOCA_APSH_LDRMODULE_WINDOWS_BASE_DLL_NAME = 1000
ldrmodule full dll name
DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE
ldrmodule size of image
DOCA_APSH_LDRMODULE_WINDOWS_INLOAD
ldrmodule appear in inload list
DOCA_APSH_LDRMODULE_WINDOWS_INMEM
ldrmodule appear in inmem list
DOCA_APSH_LDRMODULE_WINDOWS_ININIT
ldrmodule appear in ininit list

enum doca_apsh_lib_attr

doca app shield lib attributes

Values

DOCA_APSH_LIB_PID
lib pid
DOCA_APSH_LIB_COMM
lib name
DOCA_APSH_LIB_LIBRARY_PATH
lib loaded library path
DOCA_APSH_LIB_LOAD_ADRESS
lib load address for both Windows and Linux
DOCA_APSH_LIB_WINDOWS_FULL_DLL_NAME = 1000
lib full dll name
DOCA_APSH_LIB_WINDOWS_SIZE_OFIMAGE
lib size of image
DOCA_APSH_LIB_LINUX_LOAD_ADRESS = 2000
lib load address for Linux. It's kept for backwards compatibility, use DOCA_APSH_LIB_LOAD_ADRESS instead-

enum doca_apsh_module_attr

doca app shield module attributes

Values

DOCA_APSH_MODULES_OFFSET
module offset
DOCA_APSH_MODULES_NAME
module name
DOCA_APSH_MODULES_SIZE
module size

enum doca_apsh_netscan_attr

doca app shield netsacn attributes

Values

DOCA_APSH_NETSCAN_PID
netscan process id
DOCA_APSH_NETSCAN_COMM
netscan process name
DOCA_APSH_NETSCAN_PROTOCOL
netscan connection protcol
DOCA_APSH_NETSCAN_LOCAL_ADDR
netscan connection local address
DOCA_APSH_NETSCAN_REMOTE_ADDR

netscan connection remote address

DOCA_APSH_NETSCAN_LOCAL_PORT
netscan connection local port

DOCA_APSH_NETSCAN_REMOTE_PORT
netscan connection remote port

DOCA_APSH_NETSCAN_STATE
netscan connection state

DOCA_APSH_NETSCAN_TIME
netscan connection creation time

enum doca_apsh_privilege_attr

doca app shield privileges attributes windows privilege list can be found on: https://docs.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

Values

DOCA_APSH_PRIVILEGES_PID
privilege process pid

DOCA_APSH_PRIVILEGES_COMM
privilege process name

DOCA_APSH_PRIVILEGES_NAME
privilege name, for example: SeTcbPrivilege

DOCA_APSH_PRIVILEGES_IS_ON
is the privilege turned on or off. For Windows this is the outcome of get(PRESENT) && (get(ENABLED) || get(DEFAULT))

DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT = 1000
privilege present flag

DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED
privilege enabled flag

DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT
privilege enabledbydefault flag

enum doca_apsh_process_attr

doca app shield process attributes

Values

DOCA_APSH_PROCESS_PID
process id

DOCA_APSH_PROCESS_PPID
process parent id

DOCA_APSH_PROCESS_COMM
process executable name

DOCA_APSH_PROCESS_CPU_TIME

process cpu time [ps]

DOCA_APSH_PROCESS_WINDOWS_OFFSET = 1000
process offset

DOCA_APSH_PROCESS_WINDOWS_THREADS
process thread count

DOCA_APSH_PROCESS_LINUX_GID = 2000
process group id

DOCA_APSH_PROCESS_LINUX_UID
process user id

DOCA_APSH_PROCESS_LINUX_STATE
process state

enum doca_apsh_process_parameters_attr

doca app shield process-parameters attributes

Values

DOCA_APSH_PROCESS_PARAMETERS_PID
process-parameters pid

DOCA_APSH_PROCESS_PARAMETERS_CMD_LINE
process-parameters command line

DOCA_APSH_PROCESS_PARAMETERS_IMAGE_BASE_ADDR
process-parameters image base address

DOCA_APSH_PROCESS_PARAMETERS_IMAGE_FULL_PATH
process-parameters image full path

enum doca_apsh_sid_attr

doca app shield SID (secruity identifiers) attributes

Values

DOCA_APSH_PROCESS_SID_PID
SID process id

DOCA_APSH_PROCESS_SID_STRING
SID string

DOCA_APSH_PROCESS_SID_ATTRIBUTES
SID attributes flag

enum doca_apsh_system_config_attr

doca app shield configuration attributes

Values

DOCA_APSH_OS_SYMBOL_MAP
os symbol map path

DOCA_APSH_MEM_REGION
memory region path
DOCA_APSH_KPGD_FILE
kpgd file path
DOCA_APSH_VHCA_ID
vhca id
DOCA_APSH_OS_TYPE
os type
DOCA_APSH_SCAN_WIN_SIZE
yara scan window size
DOCA_APSH_SCAN_WIN_STEP
yara scan window step
DOCA_APSH_HASHTEST_LIMIT
limit of vm areas to attest
DOCA_APSH_MODULES_LIMIT
limit of modules number
DOCA_APSH_PROCESS_LIMIT
limit of processes number
DOCA_APSH_THREADS_LIMIT
limit of threads number
DOCA_APSH_LDRMODULES_LIMIT
limit of ldrmodules number on windows
DOCA_APSH_LIBS_LIMIT
limit of libs number
DOCA_APSH_VADS_LIMIT
limit of vads number
DOCA_APSH_WINDOWS_ENVARS_LIMIT
length limit of envars for windows
DOCA_APSH_HANDLES_LIMIT
limit of handles number on windows
DOCA_APSH_STRING_LIMIT
length limit of apsh_read_str

enum doca_apsh_system_os

system os types

Values

DOCA_APSH_SYSTEM_LINUX
linux
DOCA_APSH_SYSTEM_WINDOWS
windows

enum doca_apsh_thread_attr

doca app shield thread attributes

Values

DOCA_APSH_THREAD_PID
thread process id
DOCA_APSH_THREAD_TID
thread id
DOCA_APSH_THREAD_STATE
thread state
DOCA_APSH_THREAD_WINDOWS_WAIT_REASON = 1000
thread wait reason
DOCA_APSH_THREAD_WINDOWS_OFFSET
thread offset
DOCA_APSH_THREAD_LINUX_PROC_NAME = 2000
thread process name
DOCA_APSH_THREAD_LINUX_THREAD_NAME
thread name

enum doca_apsh_vad_attr

doca app shield virtual address descriptor attributes

Values

DOCA_APSH_VMA_PID
vma process id
DOCA_APSH_VMA_OFFSET
vma offset
DOCA_APSH_VMA_PROTECTION
vma protection
DOCA_APSH_VMA_VM_START
vma vm start
DOCA_APSH_VMA_VM_END
vma vm end
DOCA_APSH_VMA_PROCESS_NAME
vma process name
DOCA_APSH_VMA_FILE_PATH
vma file path
DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE = 1000
vma commit charge
DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY
vma private memory

DOCA_APSH_VMA_WINDOWS_TAG
vma pool tag

enum doca_apsh_yara_attr

doca app shield yara attributes

Values

DOCA_APSH_YARA_PID
pid of the process
DOCA_APSH_YARA_COMM
name of the process
DOCA_APSH_YARA_RULE
rule name
DOCA_APSH_YARA_MATCH_WINDOW_ADDR
virtual address of the scan window of the match
DOCA_APSH_YARA_MATCH_WINDOW_LEN
length of the scan window of the match

enum doca_apsh_yara_rule

avaiable doca app shield yara rules

Values

DOCA_APSH_YARA_RULE_HELLO_WORLD
yara rule that scans for "Hello World". Rule name is "Hello_World".
DOCA_APSH_YARA_RULE_REFLECTIVE_DLL_INJECTION
yara rule that scans for Reflective Dll Injection attack. Rule name is "Reflective_Dll_Injection".
DOCA_APSH_YARA_RULE_MIMIKATZ
yara rule that scans for Mimiaktz process running on the system. Rule name is "Mimikatz".

enum doca_apsh_yara_scan_type

doca app shield yara scan type bitmask

Values

DOCA_APSH_YARA_SCAN_VMA = 1
scan all vma tree, override all others
DOCA_APSH_YARA_SCAN_HEAP = 1<<1
scan heap vads

typedef char *DOCA_APSH_ATTESTATION_COMM_TYPE

attestation comm type

typedef uint64_t DOCA_APSH_ATTESTATION_END_ADDRESS_TYPE

attestation end address type

typedef bool DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT_TYPE

attestation hash data is present type

typedef int DOCA_APSH_ATTESTATION_MATCHING_HASHES_TYPE

attestation matching hashes type

typedef int DOCA_APSH_ATTESTATION_PAGES_NUMBER_TYPE

attestation pages number type

typedef int DOCA_APSH_ATTESTATION_PAGES_PRESENT_TYPE

attestation pages present type

typedef char *DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA_TYPE

attestation path of memory area type

typedef unsigned int DOCA_APSH_ATTESTATION_PID_TYPE

attestation pid type

typedef char *DOCA_APSH_ATTESTATION_PROTECTION_TYPE

attestation protection type

typedef uint64_t DOCA_APSH_ATTESTATION_START_ADDRESS_TYPE

attestation start address type

typedef doca_dev *DOCA_APSH_DMA_DEV_TYPE

dma dev name

typedef char *DOCA_APSH_ENVARS_COMM_TYPE

envars comm type

typedef unsigned int DOCA_APSH_ENVARS_PID_TYPE

envars pid type

typedef char *DOCA_APSH_ENVARS_VALUE_TYPE

envars value type

typedef char *DOCA_APSH_ENVARS_VARIABLE_TYPE

envars variable type

typedef uint64_t DOCA_APSH_ENVARS_WINDOWS_BLOCK_TYPE

envars windows block address type

typedef uint64_t DOCA_APSH_HANDLE_ACCESS_TYPE

handle access type

typedef char *DOCA_APSH_HANDLE_COMM_TYPE

handle comm type

typedef char *DOCA_APSH_HANDLE_NAME_TYPE

handle name type

typedef unsigned int DOCA_APSH_HANDLE_PID_TYPE

handle pid type

typedef uint64_t DOCA_APSH_HANDLE_TABLE_ENTRY_TYPE

handle table entry type

typedef char *DOCA_APSH_HANDLE_TYPE_TYPE

handle type type

typedef uint64_t DOCA_APSH_HANDLE_VALUE_TYPE

handle value type

typedef int DOCA_APSH_HASHTEST_LIMIT_TYPE

limit of vm areas to attest

typedef char *DOCA_APSH_KPGD_FILE_TYPE

kpgd file path

typedef uint64_t DOCA_APSH_LDRMODULE_BASE_ADDRESS_TYPE

ldrmodule base adress type

typedef char *DOCA_APSH_LDRMODULE_COMM_TYPE

ldrmodule comm type

typedef char *DOCA_APSH_LDRMODULE_LIBRARY_PATH_TYPE

ldrmodule library path type

typedef unsigned int DOCA_APSH_LDRMODULE_PID_TYPE

ldrmodule pid type

typedef char *DOCA_APSH_LDRMODULE_WINDOWS_BASE_DLL_NAME_

ldrmodule windows BASE dll name type

typedef bool DOCA_APSH_LDRMODULE_WINDOWS_ININIT_TYPE

ldrmodule ininit type

typedef bool DOCA_APSH_LDRMODULE_WINDOWS_INLOAD_TYPE

ldrmodule inload type

typedef bool DOCA_APSH_LDRMODULE_WINDOWS_INMEM_TYPE

ldrmodule inmem type

typedef unsignedlong DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE_TY

ldrmodule size of image type

typedef char *DOCA_APSH_LIB_COMM_TYPE

lib comm type

typedef char *DOCA_APSH_LIB_LIBRARY_PATH_TYPE

lib loaded library path type

typedef uint64_t DOCA_APSH_LIB_LINUX_LOAD_ADRESS_TYPE

lib load address for Linux

typedef uint64_t DOCA_APSH_LIB_LOAD_ADRESS_TYPE

lib load address for both Windows and Linux

typedef unsigned int DOCA_APSH_LIB_PID_TYPE

lib pid type

typedef char *DOCA_APSH_LIB_WINDOWS_FULL_DLL_NAME_TYPE

lib full dll name type

typedef unsignedlong DOCA_APSH_LIB_WINDOWS_SIZE_OFIMAGE_TYPE

lib size ofimage type

typedef int DOCA_APSH_LIBS_LIMIT_TYPE

limit of libs number

typedef char *DOCA_APSH_MEM_REGION_TYPE

memory region path

typedef int DOCA_APSH_MODULES_LIMIT_TYPE

llimit of modules number

typedef char *DOCA_APSH_MODULES_NAME_TYPE

module name type

typedef uint64_t DOCA_APSH_MODULES_OFFSET_TYPE

module offset type

typedef uint32_t DOCA_APSH_MODULES_SIZE_TYPE

module size type

typedef char *DOCA_APSH_NETSCAN_COMM_TYPE

netscan process name

typedef char *DOCA_APSH_NETSCAN_LOCAL_ADDR_TYPE

netscan connection local address

typedef uint16_t DOCA_APSH_NETSCAN_LOCAL_PORT_TYPE

netscan connection local port

typedef uint32_t DOCA_APSH_NETSCAN_PID_TYPE

netscan process id

typedef char *DOCA_APSH_NETSCAN_PROTOCOL_TYPE

netscan connection protcol

typedef char *DOCA_APSH_NETSCAN_REMOTE_ADDR_TYPE

netscan connection remote address

typedef uint16_t DOCA_APSH_NETSCAN_REMOTE_PORT_TYPE

netscan connection remote port

typedef char *DOCA_APSH_NETSCAN_STATE_TYPE

netscan connection state

typedef char *DOCA_APSH_NETSCAN_TIME_TYPE

netscan connection creation time

typedef char *DOCA_APSH_OS_SYMBOL_MAP_TYPE

os symbol map path

typedef DOCA_APSH_OS_TYPE_TYPE

os type

typedef char *DOCA_APSH_PRIVILEGES_COMM_TYPE

privilege process name

typedef bool DOCA_APSH_PRIVILEGES_IS_ON_TYPE

privilege is on type

typedef char *DOCA_APSH_PRIVILEGES_NAME_TYPE

privilege name type

typedef unsigned int DOCA_APSH_PRIVILEGES_PID_TYPE

privilege process pid

typedef bool DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT_TYPE

privilege windows enabled by default type

typedef bool DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED_TYPE

privilege windows enabled type

typedef bool DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT_TYPE

privilege windows present type

typedef char *DOCA_APSH_PROCESS_COMM_TYPE

process comm type

typedef uint64_t DOCA_APSH_PROCESS_CPU_TIME_TYPE

process cpu time type

typedef int DOCA_APSH_PROCESS_LIMIT_TYPE

limit of processes number

typedef unsigned int DOCA_APSH_PROCESS_LINUX_GID_TYPE

process gid type

typedef uint64_t DOCA_APSH_PROCESS_LINUX_STATE_TYPE

process state type

typedef unsigned int DOCA_APSH_PROCESS_LINUX_UID_TYPE

process uid type

typedef char *DOCA_APSH_PROCESS_PARAMETERS_CMD_LINE_TYPE

process-parameters command line

typedef uint64_t DOCA_APSH_PROCESS_PARAMETERS_IMAGE_BASE_ADDR

process-parameters image base address

typedef char *DOCA_APSH_PROCESS_PARAMETERS_IMAGE_FULL_PATH

process-parameters image full path

typedef unsigned int DOCA_APSH_PROCESS_PARAMETERS_PID_TYPE

process-parameters pid

typedef unsigned int DOCA_APSH_PROCESS_PID_TYPE

process pid type

typedef unsigned int DOCA_APSH_PROCESS_PPID_TYPE

process pid type

typedef uint32_t DOCA_APSH_PROCESS_SID_ATTRIBUTES_TYPE

SID attributes flag.

typedef unsigned int DOCA_APSH_PROCESS_SID_PID_TYPE

SID process id.

typedef char *DOCA_APSH_PROCESS_SID_STRING_TYPE

SID strings.

typedef uint64_t DOCA_APSH_PROCESS_WINDOWS_OFFSET_TYPE

process offset type

typedef int DOCA_APSH_PROCESS_WINDOWS_THREADS_TYPE

process threads type

typedef char *DOCA_APSH_REGEX_DEV_TYPE

regex dev name

typedef uint32_t DOCA_APSH_SCAN_WIN_SIZE_TYPE

yara scan window size

typedef uint32_t DOCA_APSH_SCAN_WIN_STEP_TYPE

yara scan window step

typedef int DOCA_APSH_STRING_LIMIT_TYPE

length limit of apsh_read_str

typedef char *DOCA_APSH_THREAD_LINUX_PROC_NAME_TYPE

thread proc name type

typedef char *DOCA_APSH_THREAD_LINUX_THREAD_NAME_TYPE

thread thread name type

typedef unsigned int DOCA_APSH_THREAD_PID_TYPE

thread pid type

typedef long DOCA_APSH_THREAD_STATE_TYPE

thread state type

typedef unsigned int DOCA_APSH_THREAD_TID_TYPE

thread tid type

typedef uint64_t DOCA_APSH_THREAD_WINDOWS_OFFSET_TYPE

thread offset type

typedef unsigned char DOCA_APSH_THREAD_WINDOWS_WAIT_REASON_TYPE

thread wait reason type

typedef int DOCA_APSH_THREADS_LIMIT_TYPE

limit of threads number

typedef int DOCA_APSH_VADS_LIMIT_TYPE

limit of vads number

typedef doca_dev_rep *DOCA_APSH_VHCA_ID_TYPE

vhca id

typedef char *DOCA_APSH_VMA_FILE_PATH_TYPE

vma file path type

typedef uint64_t DOCA_APSH_VMA_OFFSET_TYPE

vma offset type

typedef unsigned int DOCA_APSH_VMA_PID_TYPE

vma pid type

typedef char *DOCA_APSH_VMA_PROCESS_NAME_TYPE

vma file path type

typedef char *DOCA_APSH_VMA_PROTECTION_TYPE

vma protection type

typedef uint64_t DOCA_APSH_VMA_VM_END_TYPE

vma vm end type

typedef uint64_t DOCA_APSH_VMA_VM_START_TYPE

vma vm start type

typedef int DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE_TYPE

vma commit charge type

typedef int DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY_TYPE

vma private memory type

typedef char *DOCA_APSH_VMA_WINDOWS_TAG_TYPE

vma tag type

typedef int DOCA_APSH_WINDOWS_ENVARS_LIMIT_TYPE

length limit of envars for windows

typedef char *DOCA_APSH_YARA_COMM_TYPE

name of the process

typedef uint64_t DOCA_APSH_YARA_MATCH_WINDOW_ADDR_TYPE

virtual address of the scan window of the match

typedef uint64_t DOCA_APSH_YARA_MATCH_WINDOW_LEN_TYPE

length of the scan window of the match

typedef uint32_t DOCA_APSH_YARA_PID_TYPE

pid of the process

typedef char *DOCA_APSH_YARA_RULE_TYPE

rule name

2.4. Core

DOCA Buffer

DOCA Buffer Array

DOCA Buffer Inventory

DOCA Bufpool

DOCA Context

DOCA Device

DOCA DPDK

DOCA Error

DOCA Hotplug

DOCA Memory Map

DOCA RDMA BRIDGE

DOCA Sync Event

DOCA Types

2.4.1. DOCA Buffer

Core

The DOCA Buffer is used for reference data. It holds the information on a memory region that belongs to a DOCA memory map, and its descriptor is allocated from DOCA Buffer Inventory.

doca_error_t doca_buf_get_data (doca_buf *buf, void **data)

Get the buffer's data.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

data
The data of the buffer. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_get_data_len (const doca_buf *buf, size_t *data_len)

Get buffer's data length.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

data_len
The data length of the buffer. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_get_head (const doca_buf *buf, void **head)

Get the buffer's head.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

head
The head of the buffer. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_get_len (const doca_buf *buf, size_t *len)

Get the buffer's length.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

len
The length of the buffer. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_get_refcount (doca_buf *buf, uint16_t *refcount)

Get the reference count of the object.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

refcount
The number of references to the object. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_list_chain (doca_buf *list1, doca_buf *list2)

Append list2 to list1.

Parameters

list1
DOCA Buf representing list1. MUST NOT BE NULL AND MUST BE HEAD OF LIST.

list2
DOCA Buf representing list2. MUST NOT BE NULL AND MUST BE HEAD OF LIST.

Returns

DOCA_SUCCESS - always.

Description

Before: +----+ +----+ +----+ list1 -> |1 |->|2 |->|3 | +----+ +----+ +----+

+----+ +----+ list2 -> |4 |->|5 | +----+ +----+

After:

+----+ +----+ +----+ +----+ +----+ list1 -> |1 |->|2 |->|3 |->|4 |->|5 | +----+ +----+ +----+ +----+ +----+ / list2

doca_error_t doca_buf_list_is_first (const doca_buf *buf, bool *is_first)

Check if provided DOCA Buf is the first element in a linked list.

Parameters

buf
DOCA Buf element.

is_first
True if buf is the first element, false if it is not.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_list_is_last (const doca_buf *buf, bool *is_last)

Check if provided DOCA Buf is the last element in a linked list.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

is_last
True if buf is the last element, false if it is not. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_list_last (doca_buf *buf, doca_buf **last_buf)

Get last DOCA Buf in linked list.

Parameters

buf
DOCA Buf element.

last_buf
The last DOCA Buf in the linked list, which may be buf.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_list_next (doca_buf *buf, doca_buf **next_buf)

Get next DOCA Buf in linked list.

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

next_buf
The next DOCA Buf in the linked list, *next_buf will be NULL if the no other element in the list. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_buf_list_num_elements (const doca_buf *buf, uint32_t *num_elements)

Get the number of the elements in list.

Parameters

buf
DOCA Buf element. Buf must be a head of a list.

num_elements
Number of elements in list.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if the buffer is not a head of a list.

doca_error_t doca_buf_list_unchain (doca_buf *list1, doca_buf *list2)

Separate list2 from list1.

Parameters

list1
DOCA Buf representing list1. MUST NOT BE NULL.

list2
DOCA Buf representing list2, list2 should be contained in list1. list2 must be different from list1. MUST NOT BE NULL

Returns

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

- DOCA_ERROR_INVALID_VALUE - if list2 is not part of list1.

Description

Before: +----+ +----+ +----+ +----+ +----+ list1 -> |1 |->|2 |->|3 |->|4 |->|5 | +----+ +----+ +----+ +----+ +----+ / list2

After: +----+ +----+ +----+ list1 -> |1 |->|2 |->|3 | +----+ +----+ +----+

+----+ +----+ list2 -> |4 |->|5 | +----+ +----+

doca_error_t doca_buf_refcount_add (doca_buf *buf, uint16_t *refcount)

Increase the object reference count by 1.

Parameters

buf
DOCA Buf element.

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

Returns

- DOCA_ERROR_NOT_SUPPORTED

Description

> Note:
>
> This function is not supported yet.

doca_error_t doca_buf_refcount_rm (doca_buf *buf, uint16_t *refcount)

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

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

refcount
The number of references to the object before this operation took place. Can be NULL.

Returns

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

- DOCA_ERROR_NOT_PERMITTED - buf is the next element in some list.

Description

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

doca_error_t doca_buf_set_data (doca_buf *buf, void *data, size_t data_len)

Parameters

buf
DOCA Buf element. MUST NOT BE NULL.

data
Data address. MUST NOT BE NULL.

data_len
Data length.

Returns

DOCA_SUCCESS - always

Description

Set data pointer and data length

+-----------+-----+-----------------+ Before | |data | | +-----------+-----+-----------------+

__data_len__ / \ +-----+--------------+--------------+ After | |data | | +-----+--------------
+--------------+ / data

Note:

The range [data, data + data_len] must be in [head, head + len]. Otherwise undefined behaviour.

2.4.2. DOCA Buffer Array

Core

The DOCA buffer array represents an array of fixed size doca_bufs (for multiple doca_dev). Can act as a free list or direct access mode.

doca_error_t doca_buf_arr_create (doca_mmap *mmap, doca_buf_arr **buf_arr)

Allocates a doca_buf_arr.

Parameters

mmap
The mmap managing the memory chunk. Must be populated with memory chunk.
buf_arr
The newly created doca_buf_arr.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - failed to allocate a doca_buf_arr.

doca_error_t doca_buf_arr_destroy (doca_buf_arr *buf_arr)

Destroys a doca buf array instance.

Parameters

buf_arr
The doca_buf_arr to destroy

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_arr_get_gpu_handle (const doca_buf_arr *buf_arr, doca_gpu_buf_arr **gpu_buf_arr)

Retrieves the handle in the gpu memory space of a doca_buf_arr.

Parameters

buf_arr
The doca_buf_arr
gpu_buf_arr

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if doca_buf_arr is not started.

doca_error_t doca_buf_arr_set_params (doca_buf_arr *buf_arr, size_t size, uint32_t num_elem, uint32_t start_offset)

Sets the buf array params.

Parameters

buf_arr
The doca_buf_arr
size
Size in bytes of a single element (must be > 0).
num_elem
Number of elements in the doca_buf_arr (must be > 0).

start_offset
Offset from mmap start to set doca_buf_arr.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if doca_buf_arr is already started

doca_error_t doca_buf_arr_set_target_gpu (doca_buf_arr *buf_arr, doca_gpu *gpu_handler)

Configures the buf array to be created on the gpu device.

Parameters

buf_arr
The doca_buf_arr
gpu_handler
The gpu device handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if doca_buf_arr is already started

doca_error_t doca_buf_arr_start (doca_buf_arr *buf_arr)

This method enables the allocation of doca_bufs.

Parameters

buf_arr
The doca_buf_arr to start

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE -
- DOCA_ERROR_NOT_PERMITTED - if mmap is not started.
- DOCA_ERROR_NO_MEMORY - failed to allocate enough space for configuration structure

Description

> Note:
>
> Before calling this function, the mmap with which the buf array was created must be started.

doca_error_t doca_buf_arr_stop (doca_buf_arr *buf_arr)

Stops a started doca buf array.

Parameters

buf_arr
The doca_buf_arr to stop

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

2.4.3. DOCA Buffer Inventory

Core

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

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

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

Parameters

inventory
The DOCA Buf inventory. MUST NOT BE NULL AND MUST BE STARTED.
mmap
DOCA memory map structure. MUST NOT BE NULL AND MUST BE STARTED.
addr
The start address of the payload. MUST NOT BE NULL.
len
The length in bytes of the payload.

buf
Doca buf allocated and initialized with args. MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if there is no suitable memory range for the given address and length.
- DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

doca_error_t doca_buf_inventory_buf_by_args (doca_buf_inventory *inventory, doca_mmap *mmap, void *addr, size_t len, void *data, size_t data_len, doca_buf **buf)

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

Parameters

inventory
The DOCA Buf inventory. MUST NOT BE NULL AND MUST BE STARTED.
mmap
DOCA memory map structure. MUST NOT BE NULL AND MUST BE STARTED.
addr
The start address of the buffer. MUST NOT BE NULL.
len
The length in bytes of the buffer.
data
The start address of the data inside the buffer. MUST NOT BE NULL.
data_len
The length in bytes of the data.
buf
Doca buf allocated and initialized with args. MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - or if there is no suitable memory range for the given address and length.
- DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

Description

Note:

The range [data, data + data_len] must fit within [addr, addr +len]. Otherwise undefined behaviour.

doca_error_t doca_buf_inventory_buf_by_data (doca_buf_inventory *inventory, doca_mmap *mmap, void *data, size_t data_len, doca_buf **buf)

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

Parameters

inventory
The DOCA Buf inventory. MUST NOT BE NULL AND MUST BE STARTED.

mmap
DOCA memory map structure. MUST NOT BE NULL AND MUST BE STARTED.

data
The start address of the data inside the buffer. MUST NOT BE NULL.

data_len
The length in bytes of the data.

buf
Doca buf allocated and initialized with args. MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - or if there is no suitable memory range for the given address and length.
- DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

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

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

Parameters

inventory
Buffer inventory structure that will hold the new doca_buf.

src_buf
The DOCA buf to be duplicated.

dst_buf
A duplicate DOCA Buf.

Returns

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

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

doca_error_t doca_buf_inventory_create (const doca_data *user_data, size_t num_elements, uint32_t extensions, doca_buf_inventory **buf_inventory)

Allocates buffer inventory with default/unset attributes.

Parameters

user_data

num_elements
Initial number of elements in the inventory.

extensions
Bitmap of extensions enabled for the inventory described in doca_buf.h.

buf_inventory
Buffer inventory with default/unset attributes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

- DOCA_ERROR_NO_MEMORY - failed to alloc doca_buf_inventory.

Description

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

doca_error_t doca_buf_inventory_destroy (doca_buf_inventory *inventory)

Destroy buffer inventory structure.

Parameters

inventory
Buffer inventory structure.

Returns

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

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

Description

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

doca_error_t doca_buf_inventory_get_num_elements (doca_buf_inventory *inventory, uint32_t *num_of_elements)

Read the total number of elements in a DOCA Inventory.

Parameters

inventory
The DOCA Buf inventory.
num_of_elements
The total number of elements in inventory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The total number of elements type: uint32_t.

doca_error_t doca_buf_inventory_get_num_free_elements (doca_buf_inventory *inventory, uint32_t *num_of_free_elements)

Get the total number of free elements in a DOCA Inventory.

Parameters

inventory
The DOCA Buf inventory.

num_of_free_elements
The total number of free elements in inventory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The total number of free elements type: uint32_t.

doca_error_t doca_buf_inventory_get_user_data (doca_buf_inventory *inventory, doca_data *user_data)

Get the user_data of a DOCA Inventory.

Parameters

inventory
The DOCA Buf inventory.

user_data
The user_data of inventory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The user_data that was provided to the inventory upon its creation.

doca_error_t doca_buf_inventory_start (doca_buf_inventory *inventory)

Start element retrieval from inventory.

Parameters

inventory
Buffer inventory structure.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Un-started/stopped buffer inventory rejects all attempts to retrieve element. On first start verifies & finalizes the buffer inventory object configuration.

The following become possible only after start:

- Retrieval of free elements from the inventory using doca_buf_inventory_buf_by_addr().
- Duplicating a buffer's content into a buffer allocated from the inventory using doca_buf_inventory_buf_dup().

The following are NOT possible after the first time start is called:

- Setting the properties of the inventory using doca_buf_inventory_property_set().

doca_error_t doca_buf_inventory_stop (doca_buf_inventory *inventory)

Stop element retrieval from inventory.

Parameters

inventory
Buffer inventory structure.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

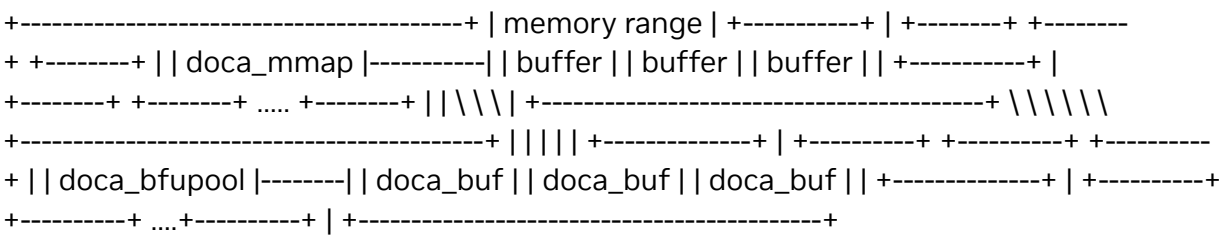
No retrieval of elements with for stopped inventory. For details see doca_buf_inventory_start().

2.4.4. DOCA Bufpool

Core

The DOCA Bufpool is an inventory of doca_buf objects, such that each doca_buf is set with a permanent, fixed size memory buffer, right from creation and till destruction, which allows immediate allocation of doca_buf objects.

Basic structure example of Bufpool (after creation):



doca_error_t doca_bufpool_buf_alloc (doca_bufpool *bufpool, doca_buf **buf)

This method acquires a doca_buf from the doca_bufpool, pointing to an allocated empty buffer.

Parameters

bufpool

The DOCA Bufpool from which to acquire a doca_buf, that was set to point to a memory buffer at doca_bufpool_create().

buf

Description

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if bufpool is un-started/stopped.
- DOCA_ERROR_NO_MEMORY - if the bufpool is empty (all doca_bufs are already allocated).

doca_error_t doca_bufpool_create (const doca_data *user_data, size_t num_elements, uint32_t extensions, size_t element_size, size_t element_alignment, const doca_mmap *mmap, doca_bufpool **bufpool)

Allocates a bufpool and sets it with doca_buf objects that are set in advance with constant memory buffers.

Parameters

user_data

num_elements

Number of elements in the bufpool (must be > 0).

extensions

Bitmap of extensions enabled for the bufpool described in doca_buf.h.

element_size

Size of a single element (must be > 0).

element_alignment

Element alignment requirement (must be a power of 2, can be 0).

mmap

The mmap managing the memory chunk. Must be populated with memory chunk.

bufpool

The newly created DOCA Bufpool.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - failed to allocate a doca_bufpool.
- DOCA_ERROR_BAD_STATE - if no memory range was set to mmap.

doca_error_t doca_bufpool_destroy (doca_bufpool *bufpool)

Destroy bufpool structure.

Parameters

bufpool
The DOCA Bufpool to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_IN_USE - if not all allocated doca_bufs had been returned to the bufpool.

Description

Before calling this function, all allocated doca_bufs should be returned back to the bufpool.

doca_error_t doca_bufpool_get_num_elements (const doca_bufpool *bufpool, uint32_t *num_of_elements)

Get the number of elements that was set in the creation of a doca_bufpool.

Parameters

bufpool
The DOCA Bufpool.
num_of_elements
The number of elements that was set in the creation of bufpool.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_bufpool_get_num_free_elements (const doca_bufpool *bufpool, uint32_t *num_of_free_elements)

Get the total number of free elements available for allocation in a doca_bufpool.

Parameters

bufpool
The DOCA Bufpool.

num_of_free_elements
The total number of free elements in bufpool.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_bufpool_get_user_data (const doca_bufpool *bufpool, doca_data *user_data)

Get the user_data of a DOCA Bufpool.

Parameters

bufpool
The DOCA Bufpool.

user_data
The user_data of bufpool.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The user_data that was provided to the bufpool upon its creation.

doca_error_t doca_bufpool_start (doca_bufpool *bufpool)

Start DOCA bufpool.

Parameters

bufpool
The DOCA Bufpool to start.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if the mmap with which the bufpool was created is not started.

Description

This method enables the allocation of doca_bufs using doca_bufpool_buf_alloc(). Before calling this function, the mmap with which the bufpool was created must be started.

doca_error_t doca_bufpool_stop (doca_bufpool *bufpool)

Stop a started DOCA bufpool.

Parameters

bufpool

The DOCA Bufpool to stop.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

This method disables the allocation of doca_bufs. Calling this method is also possible if there are allocated doca_bufs.

2.4.5. DOCA Context

Core

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

struct doca_event

Activity completion event.

struct doca_job

Job structure describes request arguments for service provided by context.

doca_error_t doca_ctx_dev_add (doca_ctx *ctx, doca_dev *dev)

Add a device to a DOCA CTX.

Parameters

ctx
The CTX to add the device to.

dev
The device to add.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is started.
- DOCA_ERROR_IN_USE - the device was already added.
- DOCA_ERROR_NOT_SUPPORTED - the provided device is not supported by CTX, I.e., the device is not useful for any job, missing the capabilities, or was opened using doca_dev_open_from_pd().
- DOCA_ERROR_DRIVER - failed to interact with device.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

doca_error_t doca_ctx_dev_rm (doca_ctx *ctx, doca_dev *dev)

Remove a device from a context.

Parameters

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

dev
The device to remove.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is started.
- DOCA_ERROR_NOT_FOUND - the provided device was never added to the ctx or was already removed.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

doca_error_t doca_ctx_get_event_driven_supported (doca_ctx *ctx, uint8_t *event_supported)

Check if CTX supports event driven mode.

Parameters

ctx
The library instance containing the WorkQ.

event_supported
Boolean indicating whether event driven mode is supported.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

In case the support exists, then this CTX can be added to WorkQ operating in event driven mode.

doca_error_t doca_ctx_get_max_num_ctx (uint32_t *max_num_ctx)

Get the ctx maximum number of contexts allowed within an application.

Parameters

max_num_ctx
The ctx max number of contexts allowed within an application.

Returns

DOCA_SUCCESS - in case max_num_ctx received the required value properly. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - max_num_ctx is NULL.

doca_error_t doca_ctx_set_datapath_on_gpu (doca_ctx *ctx, doca_gpu *gpu_dev)

This function binds the DOCA context to a gpu device.

Parameters

ctx
The library instance.

gpu_dev
A pointer to a doca_gpu device.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is started.

Description

The data path will be executed on the device and not on the CPU.

doca_error_t doca_ctx_start (doca_ctx *ctx)

Finalizes all configurations, and starts the DOCA CTX.

Parameters

ctx
The DOCA context to start.

Returns

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

- DOCA_ERROR_INVALID_VALUE - either an invalid input was received or no devices were added to the CTX.
- DOCA_ERROR_NOT_SUPPORTED - one of the provided devices is not supported by CTX.
- DOCA_ERROR_INITIALIZATION - resource initialization failed (could be due to allocation failure), or the device is in a bad state or another reason caused initialization to fail.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

Description

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

The following become possible only after start:

- Adding WorkQ to CTX using doca_ctx_workq_add()
- Removing WorkQ from CTX using doca_ctx_workq_rm()
- Submitting a job using doca_workq_submit()

The following are NOT possible after start and become possible again after calling doca_ctx_stop:

- Adding device to CTX using doca_ctx_dev_add()
- Removing device from CTX using doca_ctx_dev_rm()
- Binding gpu device to CTX using doca_ctx_set_datapath_on_gpu()

doca_error_t doca_ctx_stop (doca_ctx *ctx)

Stops the context allowing reconfiguration.

Parameters

ctx

The DOCA context to stop.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_PERMITTED - either some jobs are still pending or not all WorkQs have been removed.
- DOCA_ERROR_IN_USE - some workqs are still associated with the ctx.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

Description

Once a context has started, it can't be configured any further. This method should be called in case the context needs to be configured after starting. For more details see doca_ctx_start().

doca_error_t doca_ctx_workq_add (doca_ctx *ctx, doca_workq *workq)

Add a workQ to a context.

Parameters

ctx
The library instance that will handle the jobs.

workq
The WorkQ where you want to receive job completions.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is not started.
- DOCA_ERROR_IN_USE - same WorkQ already added.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_INITIALIZATION - initialization of WorkQ failed.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

Description

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

doca_error_t doca_ctx_workq_rm (doca_ctx *ctx, doca_workq *workq)

Remove a DOCA WorkQ from a DOCA CTX.

Parameters

ctx
The library instance containing the WorkQ.

workq
The WorkQ to remove.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

- DOCA_ERROR_BAD_STATE - CTX is not started.
- DOCA_ERROR_NOT_PERMITTED - some jobs are still pending completion.
- DOCA_ERROR_NOT_FOUND - WorkQ does not exist within CTX.
- DOCA_ERROR_IN_USE - WorkQ contains inflight jobs.
- DOCA_ERROR_UNEXPECTED - ctx is corrupted.

Description

This function can only be used after CTX is started (doca_ctx_start()).

doca_error_t doca_workq_create (uint32_t depth, doca_workq **workq)

Creates empty DOCA WorkQ object with default attributes.

Parameters

depth
The maximum number of inflight jobs.

workq
The newly created WorkQ.

Returns

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

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

Description

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

doca_error_t doca_workq_destroy (doca_workq *workq)

Destroy a DOCA WorkQ.

Parameters

workq
The WorkQ to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - invalid input received.

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

Description

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

doca_error_t doca_workq_event_handle_arm (doca_workq *workq)

Arm the WorkQ to receive next completion event.

Parameters

workq
The WorkQ object to arm. MUST NOT BE NULL.

Returns

- DOCA_SUCCESS - workq has been successfully armed, event handle can be used to wait on events.

Description

This method should be used before waiting on the event handle. The expected flow is as follows: 1. Enable event driven mode using doca_workq_set_event_driven_enable(). 2. Get event handle using doca_workq_get_event_handle(). 3. Arm the workq. 4. Wait for an event using the event handle. E.g., using epoll_wait(). 5. Once the thread wakes up, call doca_workq_event_handle_clear(). 6. Call doca_workq_progress_retrieve() until an event is retrieved. 7. Repeat 3.

Note:

event driven mode must be active by using doca_workq_set_event_driven_enable(). Otherwise undefined behaviour.

doca_error_t doca_workq_event_handle_clear (doca_workq *workq, doca_event_handle_t handle)

Clear triggered events.

Parameters

workq
The WorkQ object that received the events. MUST NOT BE NULL.
handle
workq event handle.

Returns

- DOCA_SUCCESS - on successfuly clearing triggered events.
- DOCA_ERROR_OPERATING_SYSTEM - a system call has failed.

Description

Method used for clearing of events, this method should be called after an event has been received using the event handle. After this is called, the events will no longer be triggered, and the handle can be armed again. See doca_workq_event_handle_arm() for entire flow.

Note:

event driven mode must be active by using doca_workq_set_event_driven_enable(). Otherwise undefined behaviour.

doca_error_t doca_workq_get_depth (const doca_workq *workq, uint32_t *depth)

Get the maximum number of inflight jobs allowed for a DOCA workq.

Parameters

workq
The DOCA WorkQ.

depth
The maximum number of inflight jobs allowed for workq.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_workq_get_event_driven_enable (doca_workq *workq, uint8_t *enabled)

Check if WorkQ event-driven mode is enabled.

Parameters

workq
The WorkQ to query.

enabled
0 or 1 indicating if event-driven mode is enabled.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Event-driven mode is not enabled by default. It is possible to enable it by setting this porperty to 1. Using doca_workq_set_event_driven_enable()

doca_error_t doca_workq_get_event_handle (doca_workq *workq, doca_event_handle_t *handle)

Get the event handle for waiting on events.

Parameters

workq
The WorkQ to query.

handle
The event handle of the WorkQ.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - event driven mode is not enabled. Try doca_workq_set_event_driven_enable().

Description

Calling this for the first time will enable event-driven mode for the WorkQ. Retrieves the event handle of the WorkQ, the handle does not change throughout the lifecycle of the WorkQ.

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

Progress & retrieve single pending event.

Parameters

workq
The WorkQ object to poll for events. MUST NOT BE NULL.

ev
Event structure to be filled in case an event was received. MUST NOT BE NULL
flags
Flags for progress/retrival operations. A combination of enum doca_workq_retrieve_flags.

Returns

- DOCA_SUCCESS - on successful event retrieval. ev output argument is set.
- DOCA_ERROR_AGAIN - no event available (ev output argument not set), try again to make more progress.
- DOCA_ERROR_IO_FAILED - the retrieved event is a failure event. The specific error is reported per action type.

Description

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

NOTE: for V1 retrieve supported for single event only.

doca_error_t doca_workq_set_event_driven_enable (doca_workq *workq, uint8_t enable)

Enable WorkQ event-driven mode.

Parameters

workq
The WorkQ to query.
enable
0 or 1 indicating whether to enable event-driven mode or not.

Returns

DOCA_SUCCESS - in case event driven mode has been set, or is already set to same value. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - workq is still added to at least 1 CTX.
- DOCA_ERROR_OPERATING_SYSTEM - a system call has failed.

Description

Event-driven mode is not enabled by default. Once enabled, the doca_workq_handle_* APIs can be used in order to wait on events. This mode can only be enabled before adding the WorkQ to any CTX.

doca_error_t doca_workq_set_event_handle (doca_workq *workq, doca_event_handle_t event_handle, uintptr_t completion_key)

Set event handle This method is supported only for Windows. Windows uses io completion port that is created by the application and passed to the work queue. The work queue Uses the io completion port to register events.

Parameters

workq
The work queue to set

event_handle
The IO completion port to register to

completion_key
Completion key facilitates finding the source of the event.

Returns

DOCA_SUCCESS - in case event driven mode has been set, or is already set to same value. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - used in non Windows OS.
- DOCA_ERROR_OPERATING_SYSTEM - a system call has failed.

doca_error_t doca_workq_submit (doca_workq *workq, doca_job *job)

Submit a job to a DOCA WorkQ.

Parameters

workq
The DOCA WorkQ used for progress and retrieval of jobs. MUST NOT BE NULL.

job
The job to submit, the job must be compatible with the WorkQ. MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_BAD_STATE - in case job->ctx is stopped.
- DOCA_ERROR_NOT_FOUND - in case the ctx is not associated to the workQ.
- DOCA_ERROR_NO_MEMORY - in case the workQ is full. Maximum number of inflight jobs.

Description

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

Note:

job->ctx must not be NULL, must be started and associated with the workq. Otherwise undefined behaviour.

#define DOCA_ACTION_SDK_RANGE 16

Power 2 single SDK/context action type range.

2.4.6. DOCA Device

Core

The DOCA device represents an available processing unit backed by the HW or SW implementation.

enum doca_dev_rep_filter

Representor device filter by flavor

Multiple options possible but some are mutually exclusive.

Values

DOCA_DEV_REP_FILTER_ALL = 0
DOCA_DEV_REP_FILTER_NET = 1<<1
DOCA_DEV_REP_FILTER_VIRTIO_FS = 1<<2
DOCA_DEV_REP_FILTER_VIRTIO_NET = 1<<3
DOCA_DEV_REP_FILTER_VIRTIO_BLK = 1<<4
DOCA_DEV_REP_FILTER_NVME = 1<<5

__DOCA_EXPERIMENTAL doca_devinfo *doca_dev_as_devinfo (const doca_dev *dev)

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

Parameters

dev
The doca device instance.

Returns

The matching doca_devinfo instance in case of success, NULL in case dev is invalid or was created by doca_dev_open_from_pd().

doca_error_t doca_dev_close (doca_dev *dev)

Destroy allocated local device instance.

Parameters

dev
The local doca device instance.

Returns

DOCA_SUCCESS - in case of success.

- DOCA_ERROR_IN_USE - failed to deallocate device resources.

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

Initialize local device for use.

Parameters

devinfo
The devinfo structure of the requested device.
dev
Initialized local doca device instance on success. Valid on success only.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - failed to allocate protection domain for device.
- DOCA_ERROR_NOT_CONNECTED - failed to open device.

- DOCA_ERROR_INITIALIZATION - maximum number of open devices was exceeded.

Description

Note:

In case the same device was previously opened, then the same doca_dev instance is returned.

__DOCA_EXPERIMENTAL doca_devinfo_rep *doca_dev_rep_as_devinfo (doca_dev_rep *dev_rep)

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

Parameters

dev_rep

The representor doca device instance.

Returns

The matching doca_devinfo_rep instance in case of success, NULL in case dev_rep is invalid.

doca_error_t doca_dev_rep_close (doca_dev_rep *dev)

Destroy allocated representor device instance.

Parameters

dev

The representor doca device instance.

Returns

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

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

doca_error_t doca_dev_rep_open (doca_devinfo_rep *devinfo, doca_dev_rep **dev_rep)

Initialize representor device for use.

Parameters

devinfo
The devinfo structure of the requested device.

dev_rep
Initialized representor doca device instance on success. Valid on success only.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory for device.

doca_error_t doca_devinfo_get_ibdev_name (const doca_devinfo *devinfo, char *ibdev_name, uint32_t size)

Get the name of the IB device represented by a DOCA devinfo.

Parameters

devinfo
The device to query.

ibdev_name
The name of the IB device represented by devinfo.

size
The size of the input ibdev_name buffer, must be at least DOCA_DEVINFO_IBDEV_NAME_SIZE which includes the null terminating byte.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - no memory (exception thrown).

Description

The name of the IB device type: char[DOCA_DEVINFO_IBDEV_NAME_SIZE].

doca_error_t doca_devinfo_get_iface_name (const doca_devinfo *devinfo, char *iface_name, uint32_t size)

Get the name of the ethernet interface of a DOCA devinfo.

Parameters

devinfo
The device to query.

iface_name
The name of the ethernet interface of devinfo.

size
The size of the input iface_name buffer, must be at least DOCA_DEVINFO_IFACE_NAME_SIZE which includes the null terminating byte.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the interface name from the OS

Description

The name of the ethernet interface is the same as it's name in ifconfig. The name of the ethernet interface type: char[DOCA_DEVINFO_IFACE_NAME_SIZE].

doca_error_t doca_devinfo_get_ipv4_addr (const doca_devinfo *devinfo, uint8_t *ipv4_addr, uint32_t size)

Get the IPv4 address of a DOCA devinfo.

Parameters

devinfo
The device to query.

ipv4_addr
The IPv4 address of devinfo.

size
The size of the input ipv4_addr buffer, must be at least DOCA_DEVINFO_IPV4_ADDR_SIZE

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv4 address from the OS

Description

The IPv4 address type: uint8_t[DOCA_DEVINFO_IPV4_ADDR_SIZE].

doca_error_t doca_devinfo_get_ipv6_addr (const doca_devinfo *devinfo, uint8_t *ipv6_addr, uint32_t size)

Get the IPv6 address of a DOCA devinfo.

Parameters

devinfo
The device to query.
ipv6_addr
The IPv6 address of devinfo.
size
The size of the input ipv6_addr buffer, must be at least DOCA_DEVINFO_IPV6_ADDR_SIZE

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv6 address from the OS

Description

The IPv6 address type: uint8_t[DOCA_DEVINFO_IPV6_ADDR_SIZE].

doca_error_t doca_devinfo_get_is_hotplug_manager_supported (const doca_devinfo *devinfo, uint8_t *is_hotplug_manager)

Get the hotplug manager capability of a DOCA devinfo.

Parameters

devinfo
The device to query.
is_hotplug_manager
1 if the hotplug manager capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

The hotplug manager property type: uint8_t*.

doca_error_t doca_devinfo_get_is_mmap_export_dpu_supported (const doca_devinfo *devinfo, uint8_t *mmap_export)

Get the mmap export to DPU capability of the device.

Parameters

devinfo
The device to query.

mmap_export
1 if the mmap export capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to export an mmap. See doca_mmap_export_dpu() in doca_mmap.h true - device can be used with the mmap export API. false - export API is guaranteed to faile with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_get_is_mmap_from_export_dpu_supported (const doca_devinfo *devinfo, uint8_t *from_export)

Get the mmap create from export DPU capability of the device.

Parameters

devinfo
The device to query.

from_export
1 if the mmap from export DPU capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create an mmap from an exported mmap where the exported mmap was created using doca_mmap_export_dpu(). See doca_mmap_create_from_export() in doca_mmap.h true - device can be used with the mmap create from export API. false - create from export API is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_get_lid (const doca_devinfo *devinfo, uint16_t *lid)

Get the port LID of a DOCA devinfo.

Parameters

devinfo
The device to query.

lid
The port LID of devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query port LID.

- DOCA_ERROR_NOT_SUPPORTED - the device port's link layer is not IB.

Description

The port LID type: uint16_t *.

doca_error_t doca_devinfo_get_mac_addr (const doca_devinfo *devinfo, uint8_t *mac_addr, uint32_t size)

Get the MAC address of a DOCA devinfo.

Parameters

devinfo
The device to query.

mac_addr
The MAC address of devinfo.

size
The size of the input mac_addr buffer, must be at least DOCA_DEVINFO_MAC_ADDR_SIZE

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - the device port's link layer is not RoCE.

Description

The MAC address type: uint8_t[DOCA_DEVINFO_MAC_ADDR_SIZE].

doca_error_t doca_devinfo_get_pci_addr (const doca_devinfo *devinfo, doca_pci_bdf *pci_addr)

Get the PCI address of a DOCA devinfo.

Parameters

devinfo
The device to query.

pci_addr
The PCI address of devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_CONNECTED - provided devinfo does not support this property.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the PCI address from the OS

Description

The BDF of the device - same as the address in lspci. The PCI address type: struct doca_pci_bdf.

doca_error_t doca_devinfo_list_create (doca_devinfodev_list, uint32_t *nb_devs)

Creates list of all available local devices.

Parameters

dev_list

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

nb_devs

Number of available local devices.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - failed to allocate enough space.
- DOCA_ERROR_NOT_FOUND - failed to get RDMA devices list

Description

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

Note:

Returned list must be destroyed using doca_devinfo_list_destroy()

doca_error_t doca_devinfo_list_destroy (doca_devinfo **dev_list)

Destroy list of local device info structures.

Parameters

dev_list
List to be destroyed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - at least one device in the list is in a corrupted state.

Description

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

doca_error_t doca_devinfo_rep_get_is_list_all_supported (const doca_devinfo *devinfo, uint8_t *all_supported)

Get the representor devices discovery capability of the device.

Parameters

devinfo
The device to query.
all_supported
1 if the rep list all capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of representor devices. In case true is returned, then this device supports at least one representor type. See doca_devinfo_rep_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REP_FILTER_ALL. false - providing DOCA_DEV_REP_FILTER_ALL is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_is_list_net_supported (const doca_devinfo *devinfo, uint8_t *net_supported)

Get the remote net discovery capability of the device.

Parameters

devinfo
The device to query.

net_supported
1 if the rep list net capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of net remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_NET. false - providing DOCA_DEV_REMOTE_FILTER_NET is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_is_list_nvme_supported (const doca_devinfo *devinfo, uint8_t *nvme_supported)

Get the remote nvme discovery capability of the device.

Parameters

devinfo
The device to query.

nvme_supported
1 if the list nvme capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of nvme remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_NVME. false - providing DOCA_DEV_REMOTE_FILTER_NVME is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_is_list_virtio_blk_supported (const doca_devinfo *devinfo, uint8_t *virtio_blk_supported)

Get the remote virtio blk discovery capability of the device.

Parameters

devinfo
The device to query.

virtio_blk_supported
1 if the list virtio blk capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of virtio blk remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_VIRTIO_BLK. false - providing DOCA_DEV_REMOTE_FILTER_VIRTIO_BLK is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_is_list_virtio_fs_supported (const doca_devinfo *devinfo, uint8_t *virtio_fs_supported)

Get the remote virtio fs discovery capability of the device.

Parameters

devinfo
The device to query.

virtio_fs_supported
1 if the list virtio fs capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of virtio fs remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_VIRTIO_FS. false - providing DOCA_DEV_REMOTE_FILTER_VIRTIO_FS is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_is_list_virtio_net_supported (const doca_devinfo *devinfo, uint8_t *virtio_net_supported)

Get the remote virtio net discovery capability of the device.

Parameters

devinfo
The device to query.

virtio_net_supported
1 if the list virtio net capability is supported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create list of virtio net remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_VIRTIO_NET. false - providing DOCA_DEV_REMOTE_FILTER_VIRTIO_NET is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

doca_error_t doca_devinfo_rep_get_pci_addr (const doca_devinfo_rep *devinfo_rep, doca_pci_bdf *pci_addr)

Get the PCI address of a DOCA devinfo_rep.

Parameters

devinfo_rep
The representor of device to query.

pci_addr
The PCI address of the devinfo_rep.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The PCI address type: struct doca_pci_bdf.

doca_error_t doca_devinfo_rep_get_pci_func_type (const doca_devinfo_rep *devinfo_rep, doca_pci_func_type * *pci_func_type)

Get the PCI function type of a DOCA devinfo_rep.

Parameters

devinfo_rep
The representor of device to query.

pci_func_type
The PCI function type of the devinfo_rep.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The pci function type: enum doca_pci_func_type.

doca_error_t doca_devinfo_rep_get_vuid (const doca_devinfo_rep *devinfo_rep, char *rep_vuid, uint32_t size)

Get the Vendor Unique ID of a representor DOCA devinfo.

Parameters

devinfo_rep
The representor device to query.

rep_vuid
The Vendor Unique ID of devinfo_rep.

size
The size of the vuid buffer, including the terminating null byte ('').

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The Vendor Unique ID is used as stable ID of a VF/PF. The Vendor Unique ID type: char[DOCA_DEVINFO_VUID_SIZE].

doca_error_t doca_devinfo_rep_list_create (doca_dev *dev, int filter, doca_devinfo_repdev_list_rep, uint32_t *nb_devs_rep)

Create list of available representor devices accessible by dev.

Parameters

dev
Local device with access to representors.

filter
Bitmap filter of representor types. See enum doca_dev_rep_filter for more details.
dev_list_rep
Pointer to array of pointers. Output can then be accessed as follows (*dev_list_rep) [idx].
nb_devs_rep
Number of available representor devices.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory for list.
- DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_dev_open_from_pd()

Description

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

Note:

Returned list must be destroyed using doca_devinfo_rep_list_destroy()

doca_error_t doca_devinfo_rep_list_destroy (doca_devinfo_rep **dev_list_rep)

Destroy list of representor device info structures.

Parameters

dev_list_rep
List to be destroyed.

Returns

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

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

Description

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

2.4.7. DOCA DPDK

Core

DOCA API for integration with DPDK.

doca_error_t doca_dpdk_mempool_create (const rte_mempool *mbuf_pool, doca_dpdk_mempool **mempool_out)

Create a DOCA DPDK memory pool, with ability to convert rte_mbuf to doca_buf Expected flow is as follows: Control path: // Create the memory pool based on a DPDK memory pool doca_dpdk_mempool_create() // Add 1 or more DOCA devices doca_dpdk_mempool_dev_add() // Set permission level across all devices (default=LOCAL_READ/WRITE) doca_dpdk_mempool_set_permissions() // Start the pool doca_dpdk_mempool_start().

Parameters

mbuf_pool
A DPDK pool of mbufs, created with rte_pktmbuf_pool_create*()

mempool_out
The newly created DOCA DPDK memory pool in case of success

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Data path: // Convert DPDK mbuf to DOCA buf doca_dpdk_mempool_mbuf_to_buf() // Optionally release DPDK mbuf back to the DPDK pool in case it is no longer needed rte_pktmbuf_free() // Release the doca_buf once finished with it doca_buf_refcnt_rm()

doca_error_t doca_dpdk_mempool_destroy (doca_dpdk_mempool *mempool)

Destroy a DOCA DPDK memory pool Before destroying need to make sure that all buffers that were acquired using doca_dpdk_mempool_mbuf_to_buf() have been released This must be called before destroying the originating DPDK mempool.

Parameters

mempool
The DOCA DPDK memory pool to destroy

Returns

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

- DOCA_ERROR_IN_USE - at least 1 DOCA buf has been acquired and still not released

Description

Note:

: Once destroyed the originating DPDK memory pool, and any allocated RTE mbuf are not affected

doca_error_t doca_dpdk_mempool_dev_add (doca_dpdk_mempool *mempool, doca_dev *dev)

Add a DOCA device to the mempool This allows the DOCA bufs that are retrieved from the pool to be compatible with other DOCA libraries, that use the DOCA device.

Parameters

mempool
The DOCA DPDK memory pool to add the device to

dev
A DOCA device instance

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NO_MEMORY - out of memory.

Description

Note:

Once device has been added it can't be removed. Only option is to destroy the doca_dpdk_mempool

doca_error_t doca_dpdk_mempool_mbuf_to_buf (doca_dpdk_mempool *mempool, doca_buf_inventory *inventory, rte_mbuf *mbuf, doca_buf **buf)

Acquire a doca_buf based on an rte_mbuf The acquired doca_buf attempts to be as similar as possible to the rte_mbuf Level of support:After acquiring the buffer the refcount of the mbuf is increasedIn case mbuf is indirect refcount of the direct buffer is increased instead and metadata of the indirectmbuf is used where metdata refers to the mbuf's data offset, data length, and next pointerIn case the acquired doca_buf is duplicated, then the duplication process will increase the refcount of the direct mbufs as well Limitations:The mbuf must represent memory from the originating rte_mempool associated with this mempool and mbuf cannot be created from external memoryAny changes made to the rte_mbuf after the acquisition will not affect the doca_bufAny changes made to the doca_buf after acquisition will not affect the rte_mbuf.

Parameters

mempool
The DOCA DPDK memory pool created using the rte_mempool that created the rte_mbuf

inventory
A DOCA Buffer Inventory to be used for allocating the doca_buf. Must be started and have enough space

mbuf
A DPDK buffer that references memory that is within the RTE mempool associated with the DOCA DPDK mempool

buf
A DOCA buffer that references the same memory as the provided mbuf

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NO_MEMORY - The inventory does not have enough free elements.

Description

buf_addr __data_len__ \ / \ +----------+-------------+----------+ +----------+-------------+----------
+ rte_mbuf chain: | headroom | data | tailroom | --next--> | headroom | data | tailroom |
+----------+-------------+----------+ +----------+-------------+----------+

doca_buf created after calling this method:

head __data_len__ \ / \ +----------+-------------+----------+ +----------+-------------+----------
+ doca_buf list: | | data | | --next--> | | data | | +----------+-------------+----------+ +----------
+-------------+----------+

Note:

: Destroying the doca_buf using 'doca_buf_refcount_rm()' will call 'rte_pktmbuf_free_seg()' on each direct mbuf

doca_error_t doca_dpdk_mempool_set_permissions (doca_dpdk_mempool *mempool, uint32_t access_mask)

Set the read/write permissions of the memory for devices Default: DOCA_ACCESS_LOCAL_READ_WRITE Setting the permission will set the access that the added devices have over the memory of the DOCA buffers.

Parameters

mempool

The DOCA DPDK memory pool

access_mask

The access permissions - see 'enum doca_access_flags'

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input or bad access flag combination.

Description

Note:

: setting DOCA_ACCESS_DPU_* flags is invalid

doca_error_t doca_dpdk_mempool_start (doca_dpdk_mempool *mempool)

Start the DOCA DPDK memory pool Operations that must be done before start: Adding at least 1 device - doca_dpdk_mempool_dev_add() Optionally, setting the permission level - doca_dpdk_mempool_set_permissions() Operations that are allowed after start: Acquiring a matching doca_buf from an rte_mbuf - doca_dpdk_mempool_mbuf_to_buf() Destroying the DOCA DPDK memory pool - doca_dpdk_mempool_destroy().

Parameters

mempool
The DOCA DPDK memory pool to add the device to

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NO_MEMORY - out of memory.

doca_error_t doca_dpdk_port_as_dev (uint16_t port_id, doca_dev **dev)

Return the DOCA device associated with a DPDK port.

Parameters

port_id
The DPDK port identifier to get the associated DOCA device for.

dev
The DPDK DOCA device associated with the given DPDK port identifier.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NOT_FOUND - in case there is no such DPDK port associated with a DOCA device.

doca_error_t doca_dpdk_port_probe (doca_dev *dev, const char *devargs)

Attach a DPDK port specified by DOCA device.

Parameters

dev
DOCA device to attach PDK port for.

devargs
DPDK devargs style - must NOT contains the device's PCI address ([domain:]bus:devid.func).

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_DRIVER - in case of DPDK error during DPDK port attach.
- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

Description

Thread unsafe API.

It's the user responsibility to set the DPDK EAL initialization to skip probing the PCI device associated with the given DOCA device to prevent EAL from using it.

No initialization is done for the probed PDPK port and the port is not started.

2.4.8. DOCA Error

Core

DOCA Error provides information regarding different errors caused while using the DOCA libraries.

const __DOCA_EXPERIMENTAL char *doca_get_error_name (doca_error_t error)

Returns the string representation of an error code name.

Parameters

error
- Error code to convert to string.

Returns

char* pointer to a NULL-terminated string.

Description

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

const __DOCA_EXPERIMENTAL char *doca_get_error_string (doca_error_t error)

Returns the description string of an error code.

Parameters

error

- Error code to convert to description string.

Returns

char* pointer to a NULL-terminated string.

Description

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

#define DOCA_ERROR_PROPAGATE do { \ if (r == DOCA_SUCCESS) \ r = t; \ } while(0);

Save the first encountered doca_error_t.

Updates the return value variable r to hold the first error that we encountered.

#define DOCA_IS_ERROR doca_unlikely((r) != DOCA_SUCCESS)

Compiler optimized macro to check if we have an error.

Used in cases where error is unlikely to happen.

2.4.9. DOCA Hotplug

Core

DOCA API for hot plug/un-plug devices.

doca_error_t doca_dev_rep_hotplug (const doca_dev_hotplug_attr *attr, doca_dev_rep **dev_rep)

Hotplug and initialize representor device for use.

Parameters

attr
DOCA hotplug attr with designated characteristics.

dev_rep
Initialized representor doca device instance on success. Valid on success only.

Returns

DOCA_SUCCESS - in case of success. DOCA_ERROR_INVALID_VALUE - in case of invalid input. DOCA_ERROR_NOT_SUPPORTED - in case of one of non hotplug manager device or unsupported emulated device type. DOCA_ERROR_NOT_FOUND - in case the hotplugged device was exposed to the host PCI but its representor on the DPU couldn't be found.

doca_error_t doca_dev_rep_hotunplug (doca_dev_rep *rep_dev)

Destroy and unplug representor device instance.

Parameters

rep_dev
The previously hotplugged representor doca device instance.

Returns

DOCA_SUCCESS - in case of success. DOCA_ERROR_INVALID_VALUE - in case of invalid input. DOCA_ERROR_NOT_SUPPORTED - in case of static emulated representor device.

Description

Note:

For virtio representor devices it's recommended (due to a bug in Linux virtio drivers) to destroy a controller with a special preparation for hotunplug operation prior calling this function. See DOCA virtio documentation for more details.

2.4.10. DOCA Memory Map

Core

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

typedef void (doca_mmap_memrange_free_cb_t)

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

doca_error_t doca_mmap_create (const doca_data *user_data, doca_mmap **mmap)

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

Parameters

user_data

mmap

DOCA memory map structure with default/unset attributes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_mmap.

Description

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

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

doca_error_t doca_mmap_create_from_export (const doca_data *user_data, const void *export_desc, size_t export_desc_len, doca_dev *dev, doca_mmap **mmap)

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

Parameters

user_data

export_desc

An export descriptor generated by doca_mmap_export_*.

export_desc_len

Length in bytes of the export_desc.

dev
A local device connected to the device that resides in the exported mmap. In case the 'export_desc' was created using doca_mmap_export_dpu(), then device must have from export DPU capability. See doca_devinfo_get_is_mmap_from_export_dpu_supported() in doca_dev.h

mmap
DOCA memory map granting access to remote memory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or internal error. The following errors are internal and will occur if failed to produce new mmap from export descriptor:
- DOCA_ERROR_NO_MEMORY - if internal memory allocation failed.
- DOCA_ERROR_NOT_SUPPORTED - device missing create from export capability, or was opened using doca_dev_open_from_pd().
- DOCA_ERROR_NOT_PERMITTED
- DOCA_ERROR_DRIVER

Description

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

The following are NOT possible for the mmap created from export:

- Setting the properties of the mmap using doca_mmap_set_*().
- Adding a device to the mmap using doca_mmap_dev_add().
- Removing a device to the mmap using doca_mmap_dev_rm().
- Exporting the mmap using doca_mmap_export_*.

Note:

: The created object not backed by local memory.

doca_error_t doca_mmap_destroy (doca_mmap *mmap)

Destroy DOCA Memory Map structure.

Parameters

mmap
The DOCA memory map structure.

Returns

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

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

Description

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

doca_error_t doca_mmap_dev_add (doca_mmap *mmap, doca_dev *dev)

Register DOCA memory map on a given device.

Parameters

mmap
DOCA memory map structure.

dev
DOCA Dev instance with appropriate capability.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if memory deregistration failed or the operation is not premitted for the given mmap (see details in this function description).
- DOCA_ERROR_NO_MEMORY - if reached to DOCA_MMAP_MAX_NUM_DEVICES.
- DOCA_ERROR_IN_USE - if doca_dev already exists in doca_mmap.
- DOCA_ERROR_NOT_SUPPORTED - if dev was opened using doca_dev_open_from_pd().

Description

This operation is not permitted for:

- started memory map object.
- memory map object that have been exported or created from export.

doca_error_t doca_mmap_dev_rm (doca_mmap *mmap, doca_dev *dev)

Deregister given device from DOCA memory map.

Parameters

mmap
DOCA memory map structure.

dev
DOCA Dev instance that was previously added.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or doca_dev doesn't exists in doca_mmap.
- DOCA_ERROR_NOT_PERMITTED - if memory deregistration failed or the operation is not premitted for the given mmap (see details in this function description).

Description

This operation is not permitted for:

- started memory map object.
- memory map object that have been exported or created from export.

doca_error_t doca_mmap_export_dpu (doca_mmap *mmap, const doca_dev *dev, const void **export_desc, size_t *export_desc_len)

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

Parameters

mmap
DOCA memory map structure.

dev
Device previously added to the memory map via doca_mmap_dev_add(). Device must have export capability. See doca_devinfo_get_is_mmap_export_dpu_supported() in doca_dev.h

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or device does not exists in mmap.
- DOCA_ERROR_NOT_PERMITTED - the operation is not premitted for the given mmap, see details in this function description. The following errors will occur if failed to produce export descriptor:
- DOCA_ERROR_NO_MEMORY - if failed to alloc memory for export_desc.
- DOCA_ERROR_NOT_SUPPORTED - device missing export capability.
- DOCA_ERROR_DRIVER

Description

Once this function called on the object it considered as exported. The same mmap can be exported using different devices. Once mmap is stopped then any mmap created from export will be invalidated, and the 'export_desc' is destroyed.

This operation is not permitted for:

- un-started/stopped memory map object.
- memory map object that have been created from export.
- memory map with no DPU access permission set - see doca_mmap_set_permissions()

doca_error_t doca_mmap_export_rdma (doca_mmap *mmap, const doca_dev *dev, const void **export_desc, size_t *export_desc_len)

Compose memory map representation for later import with doca_mmap_create_from_export() for one of the devices previously added to the memory map. The imported mmap can then be used for RDMA operations.

Parameters

mmap
DOCA memory map structure.

dev
Device previously added to the memory map via doca_mmap_dev_add().
export_desc
On successful return should have a pointer to the allocated blob containing serialized representation of the memory map object for the device provided as `dev`.
export_desc_len
Length in bytes of the export_desc.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or device does not exists in mmap.
- DOCA_ERROR_NOT_PERMITTED - the operation is not premitted for the given mmap, see details in this function description. The following errors will occur if failed to produce export descriptor:
- DOCA_ERROR_NO_MEMORY - if failed to alloc memory for export_desc.
- DOCA_ERROR_NOT_SUPPORTED - device missing export capability, or was opened using doca_dev_open_from_pd().
- DOCA_ERROR_DRIVER

Description

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

This operation is not permitted for:

- un-started/stopped memory map object.
- memory map objects that have been created from export.
- memory map with no RDMA access permission set - see doca_mmap_set_permissions()

doca_error_t doca_mmap_get_exported (doca_mmap *mmap, uint8_t *exported)

Get the flag indicating if a DOCA Memory Map had been exported.

Parameters

mmap
The DOCA memory map structure.
exported
1 if mmap had been exported, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_get_from_export (doca_mmap *mmap, uint8_t *from_export)

Get the flag indicating if a DOCA Memory Map had been created from an export.

Parameters

mmap
The DOCA memory map structure.

from_export
1 if mmap had been created from export, 0 otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_get_max_num_devices (doca_mmap *mmap, uint32_t *max_num_devices)

Get the max number of devices to add to a DOCA Memory Map.

Parameters

mmap
The DOCA memory map structure.

max_num_devices
The max number of devices that can be added add to mmap.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_get_memrange (const doca_mmap *mmap, void **addr, size_t *len)

Get the memory range of DOCA memory map.

Parameters

mmap
DOCA memory map structure.

addr
Start address of the memory range previously set.

len
The size of the memory range in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - memrange was never set.

doca_error_t doca_mmap_get_num_bufs (doca_mmap *mmap, uint32_t *num_bufs)

Get the Total number of `struct doca_buf` objects pointing to the memory in a DOCA Memory Map.

Parameters

mmap
The DOCA memory map structure.

num_bufs
The total number of `struct doca_buf` objects pointing to the memory in mmap.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_get_user_data (doca_mmap *mmap, doca_data *user_data)

Get the user_data of a DOCA Memory Map.

Parameters

mmap
The DOCA memory map structure.

user_data
The user_data of mmap.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Note:

The user_data that was provided to the mmap upon its creation.

doca_error_t doca_mmap_set_dmabuf_memrange (doca_mmap *mmap, int dmabuf_fd, void *addr, size_t len)

Set the memory range of DOCA memory map using dmabuf.

Parameters

mmap
DOCA memory map structure.
dmabuf_fd
File descriptor of the dmabuf.
addr
Start adddress of the dmabuf.
len
The size of the memory range in bytes.

Returns

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

- DOCA_ERROR_NOT_SUPPORTED - if not called from linux operating system
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received, or addr + len overflows.
- DOCA_ERROR_BAD_STATE - if mmap is started.
- DOCA_ERROR_NOT_PERMITTED - if mmap memory range was set before

Description

This operation is not permitted for:

- started memory map object.

- memory map object that have been exported or created from export.

Note:

: this property is mandatory and can be done only once. it is only supported when used on linux operating system

doca_error_t doca_mmap_set_free_cb (doca_mmap *mmap, doca_mmap_memrange_free_cb_t *free_cb, void *opaque)

Set callback that will free the memory range when destroying DOCA memory map.

Parameters

mmap
DOCA memory map structure.

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

opaque
User opaque value passed to free_cb.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_BAD_STATE - if mmap is started.

Description

Note:

Callback is called on mmap destroy, only in case the mmap was started and destroyed without changing the callback.

doca_error_t doca_mmap_set_max_num_devices (doca_mmap *mmap, uint32_t max_num_devices)

Set a new max number of devices to add to a DOCA Memory Map.

Parameters

mmap
The DOCA memory map structure.

max_num_devices
The new max number of devices that can be added add to mmap.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if trying to set the max number of devices after first start of the mmap.

doca_error_t doca_mmap_set_memrange (doca_mmap *mmap, void *addr, size_t len)

Set the memory range of DOCA memory map.

Parameters

mmap
DOCA memory map structure.
addr
Start address of the memory range to be set.
len
The size of the memory range in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received, or addr + len overflows.
- DOCA_ERROR_BAD_STATE - if mmap is started.
- DOCA_ERROR_NOT_PERMITTED - if mmap memory range was set before

Description

This operation is not permitted for:

- started memory map object.
- memory map object that have been exported or created from export.

Note:

: this property is mandatory and can be done only once

doca_error_t doca_mmap_set_permissions (doca_mmap *mmap, uint32_t access_mask)

Set access flags of the registered memory.

Parameters

mmap
The DOCA memory map structure.

access_mask
bitwise combination of access flags - see enum doca_access_flags

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or trying to set an undefined access flag, or invalid combination
- DOCA_ERROR_BAD_STATE - If mmap is started

Description

this defines what kind of access the added devices have to the memory defined in mmap

doca_error_t doca_mmap_start (doca_mmap *mmap)

Start DOCA Memory Map.

Parameters

mmap
DOCA memory map structure.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if memory allocation failed.
- DOCA_ERROR_NOT_PERMITTED - if mmap is exported or created from export.

Description

Allows execution of different operations on the mmap, detailed below. On start verifies & finalizes the mmap object configuration.

The following become possible only after start:

- Exporting the mmap using doca_mmap_export_*.

- Mapping doca_buf structures to the memory ranges in the using doca_buf_inventory_buf_by_addr() or doca_buf_inventory_buf_dup().

The following are NOT possible while mmap is started:

- Setting the properties of the mmap through doca_mmap_set_*.
- Adding a device to the mmap using doca_mmap_dev_add().
- Removing a device to the mmap using doca_mmap_dev_rm().

doca_error_t doca_mmap_stop (doca_mmap *mmap)

Stop DOCA Memory Map.

Parameters

mmap
DOCA memory map structure.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if mmap was exported or created from export, or buffers that were created for this mmap, are still not destroyed.

Description

Prevents execution of different operations and allows operations that were available before start. For details see doca_mmap_start(). Frees any export descriptor received from doca_mmap_export_*, and invalidates any mmap created from this mmap export.

2.4.11. DOCA RDMA BRIDGE

Core

DOCA RDMA bridge.

doca_error_t doca_buf_get_mkey (const doca_buf *buf, doca_dev *dev, uint32_t *mkey)

Get lkey with doca_access_flags access for a DOCA buffer of a DOCA device.

Parameters

buf
The DOCA buffer to get lkey for. MUST NOT BE NULL.

dev
The DOCA device to get lkey for. MUST NOT BE NULL.
mkey
The returned MKey. MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if cannot find mkey by the given device.
- DOCA_ERROR_NOT_SUPPORTED - if the given access flags is not supported

Description

Note:

Access of mkey is defined by the mmap where buf was created.

doca_error_t doca_dev_get_pd (const doca_dev *dev, ibv_pd **pd)

Get the protection domain associated with a DOCA device.

Parameters

dev
DOCA device to get the pd from.
pd
The protection-domain associated with the given DOCA device.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_BAD_STATE - in case the device's pd is not valid (bad state)

doca_error_t doca_dev_open_from_pd (ibv_pd *pd, doca_dev **dev)

Open a DOCA device using an ibv_pd.

Parameters

pd
A protection domain that is not associated with any DOCA device

dev
A newly created DOCA device with same protection domain as 'pd'

Returns

DOCA_SUCCESS - in case of success

Description

Always prefer using a DOCA device obtained from doca_devinfo_list_create() This call will fail if PD was acquired by DOCA through doca_devinfo_list_create() and then doca_dev_get_pd()

This API should be used only to bridge between rdma-core and DOCA, to allow them to share memory registrations E.g., application already has logic that utilizes an ibv_pd, to read and write memory using RDMA, and wants to extend the logic by using libraries in DOCA, but such libraries will require a doca_dev and doca_buf instead of an ibv_pd and mkey in order to read write same memory. Then this method can be used to get a doca_dev that can be added to a doca_mmap, such that any doca_buf created from the doca_mmap can yield mkeys that are associated with the same ibv_pd using doca_buf_get_mkey()

For reference: doca_dev - is parallel to an ibv_pd doca_buf - is parallel to an ibv_mr registered on multiple devices doca_mmap - is parallel to creating an ibv_mr for multiple devices

The only APIs that are supported for the newly created device:

- doca_dev_close()
- doca_buf_get_mkey()
- doca_dev_get_pd()

2.4.12. DOCA Sync Event

Core

DOCA Sync Event DOCA Sync Event is a software synchronization mechanism of parallel execution across the CPU, DPU, DPA, and GPU. It is an abstraction around 64-bit value which can be updated, read, and waited upon from any of these units to achieve synchronization between executions on them.

struct doca_sync_event_job_get

struct doca_sync_event_job_update_add

struct doca_sync_event_job_update_set

struct doca_sync_event_job_wait

struct doca_sync_event_result

enum doca_sync_event_job_types

Sync Event job types for performing operation on a Sync Event thorugh DOCA job submission on a DOCA WorkQ

Values

DOCA_SYNC_EVENT_JOB_WAIT_GT = DOCA_ACTION_SYNC_EVENT_FIRST+1
Wait for the Sync Event to be grater than some value

DOCA_SYNC_EVENT_JOB_GET
Get the value of the Sync Event

DOCA_SYNC_EVENT_JOB_UPDATE_SET
Set the Sync Event value with some value

DOCA_SYNC_EVENT_JOB_UPDATE_ADD
Increment atomically the Sync Event value by some value

typedef uint64_t doca_dpa_dev_sync_event_t

DPA sync event handle type definition

typedef uint64_t doca_gpu_dev_sync_event_t

GPU sync event handle type definition

__DOCA_EXPERIMENTAL doca_ctx *doca_sync_event_as_ctx (doca_sync_event *event)

Convert a Sync Event to a DOCA context.

Parameters

event
The doca_sync_event to be converted

Returns

The matching doca_ctx instance in case of success, NULL otherwise.

Description

Set the Sync Event to operate as a DOCA Context only, hence it can be interacted with through the supported DOCA Context API.

Sync Event CTX supports the following operations: start/stop/workq_add/workq_rm/get_event_driven_supported. A device can't be attached to a sync event ctx.

A user can use an attached (to Sync Event CTX) DOCA WorkQ to perform operations on the underlying Sync Event asynchronously by submitting jobs to the attached DOCA WorkQ(see enum doca_sync_event_job_types).

It is suggested to use Sync Event in this mode to wait on a Sync Event in a blocking manner.

doca_error_t doca_sync_event_create (doca_sync_event **event)

Create a Sync Event handle.

Parameters

event

The created doca_sync_event instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_sync_event.

Description

Creates CPU handle - Host CPU or DPU's CPU.

doca_error_t doca_sync_event_create_from_export (doca_dev *dev, const uint8_t *data, size_t sz, doca_sync_event **event)

Create a Sync Event handle from an export.

Parameters

dev
doca_dev instance to be attached to the create doca_sync_event.

data
Exported doca_sync_event data stream, created by doca_sync_event_export_to_* call.

sz
Size of exported doca_sync_event data stream, created by doca_sync_event_export_to_* call.

event
The created doca_sync_event instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided doca_dev does not support creating Sync Event from export.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_sync_event.

Description

Creates a DPU handle. The DOCA Device should be capable of importing an exported Sync Event (see doca_sync_event_get_create_from_export_supported capability).

Note:

The Sync Event can only be configured and exported by the exporting process.

doca_error_t doca_sync_event_destroy (doca_sync_event *event)

Destroy a Sync Event instance.

Parameters

event
doca_sync_event to be destroyed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.

doca_error_t doca_sync_event_export_remote (doca_sync_event *event, doca_sync_event_remote_t *handle)

Obtain an handle for interacting with the associated DOCA Syn Event from a remote node.

Parameters

event
Target doca_sync_event instance to export for remote.

handle
The remote handle associated with the given doca_sync_event instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_export_to_dpa (doca_sync_event *event, doca_dpa *dpa, doca_dpa_dev_sync_event_t *dpa_dev_se_handle)

Export Sync Event to be shared with the DPA.

Parameters

event
Target doca_sync_event instance to export.

dpa
The associated DOCA DPA Context.

dpa_dev_se_handle
DOCA DPA device sync event handle that can be passed to a kernel.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this Sync Event action.

- DOCA_ERROR_NO_MEMORY - failed to alloc doca_dpa_sync_event.

Description

Create Sync Event DPA handle used for synchronize between the x86 CPU HOST and the DPA. Sync Event should be properly configured, either subscriber or publisher should be declared as DPA location. The underlying DOCA Device should be capable of exporting to DPA (see doca_sync_event_get_export_to_dpa_supported capability). A Sync Event can be exported from the Host CPU only.

The DOCA DPA Sync Event is an handle to be used from the DPA to perform operations on the associated Sync Event.

doca_error_t doca_sync_event_export_to_dpu (doca_sync_event *event, doca_dev *dev, const uint8_t **data, size_t *sz)

Export Sync Event to be shared with the DPU.

Parameters

event
Target doca_sync_event instance to export.

dev
Target dev to export.

data
The created export data stream.

sz
Size of created export data stream.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this Sync Event action.
- DOCA_ERROR_NO_MEMORY - failed to alloc data stream.

Description

Create export data stream used for synchronize between the x86 CPU HOST to DPU ARM. Sync Event should be properly configured, both subscriber and publisher must be declared as either CPU or DPU location. The underlying DOCA Device should be capable of exporting to DPU (see doca_sync_event_get_export_to_dpu_supported capability). A Sync Event can be exported from the Host CPU only.

The exported data stream an be used from the DPU to created an exported Sync Event (see doca_sync_event_create_from_export).

doca_error_t doca_sync_event_export_to_gpu (doca_sync_event *event, doca_gpu *gpu, doca_gpu_dev_sync_event_t *gpu_dev_se)

Export Sync Event to be shared with the GPU.

Parameters

event
Target doca_sync_event instance to export.

gpu
The associated DOCA GPU Context.

gpu_dev_se
DOCA GPU device sync event handle that can be passed to a kernel.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this Sync Event action.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_gpu_sync_event.

Description

Create Sync Event GPU handle used for synchronize between the x86 CPU HOST and the DPA. Sync Event should be properly configured, either subscriber or publisher should be declared as GPU location. The underlying DOCA Device should be capable of exporting to GPU (see doca_sync_event_get_export_to_gpu_supported capability). A Sync Event can be exported from the Host CPU only.

The DOCA GPU Sync Event is an handle to be used from the GPU to perform operations on the associated Sync Event.

doca_error_t doca_sync_event_get (doca_sync_event *event, uint64_t *value)

Get the value of a Sync Event synchronously.

Parameters

event
Target doca_sync_event instance to read its value.

value
The returned doca_sync_event value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_get_create_from_export_supported (const doca_devinfo *devinfo)

Parameters

devinfo
The DOCA device information.

Returns

DOCA_SUCCESS - in case device supports creating Sync Event from an export. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support importing an exported Sync Event.

Description

Check if given device is capable of creating Sync Event from an export.

doca_error_t doca_sync_event_get_export_to_dpa_supported (const doca_devinfo *devinfo)

Parameters

devinfo
The DOCA device information.

Returns

DOCA_SUCCESS - in case device supports exporting an associated Sync Event to DPA. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support exporting an associated Sync Event to DPA.

Description

Check if a DOCA device is capable of exporting an associated Sync Event to the DPA using doca_sync_event_export_to_dpa.

doca_error_t doca_sync_event_get_export_to_dpu_supported (const doca_devinfo *devinfo)

Parameters

devinfo
The DOCA device information.

Returns

DOCA_SUCCESS - in case device supports exporting an associated Sync Event to DPU. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support exporting an associated Sync Event to DPU.

Description

Check if a DOCA device is capable of exporting an associated Sync Event to the DPU using doca_sync_event_export_to_dpu.

doca_error_t doca_sync_event_get_export_to_gpu_supported (const doca_devinfo *devinfo)

Parameters

devinfo
The DOCA device information.

Returns

DOCA_SUCCESS - in case device supports exporting an associated Sync Event to GPU. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support exporting an associated Sync Event to GPU.

Description

Check if a DOCA device is capable of exporting an associated Sync Event to the GPU using doca_sync_event_export_to_gpu.

doca_error_t doca_sync_event_job_get_supported (doca_devinfo *devinfo, doca_sync_event_job_types type)

Check if a given device is capable of executing a specific Sync Event job.

Parameters

devinfo
The DOCA device information.

type
doca_sync_event_job_types available for this device. see enum doca_sync_event_job_types.

Returns

DOCA_SUCCESS - in case device supports the Sync Event job type. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this Sync Event job.

doca_error_t doca_sync_event_publisher_add_location_cpu (doca_sync_event *event, doca_dev *dev)

Associate a CPU device context as the Sync Event Publisher.

Parameters

event
Target doca_sync_event instance to set.

dev
doca_dev instance associated with CPU.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_publisher_add_location_dpa (doca_sync_event *event, doca_dpa *dpa)

Associate a DOCA DPA context as the Sync Event Publisher.

Parameters

event
Target doca_sync_event instance to set.

dpa
doca_dpa instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_publisher_add_location_dpu (doca_sync_event *event)

Declare Sync Event Publisher as the DPU.

Parameters

event
Target doca_sync_event instance to set.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.

doca_error_t doca_sync_event_publisher_add_location_gpu (doca_sync_event *event, doca_gpu *gpu)

Associate a DOCA GPU context as the Sync Event Publisher.

Parameters

event
Target doca_sync_event instance to set.

gpu
doca_gpu instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_set_addr (doca_sync_event *event, uint64_t *addr)

Set the 64-bit value's address for a Sync Event.

Parameters

event
Pointer to se event instance to be configured.

addr
Allocated address pointer.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - setting address for event which has already been started is not allowed.
- DOCA_ERROR_NOT_SUPPORTED - addr is in unsupported address space.

Description

Setting external address is allowed only for CPU/DPU configured Sync Event.

doca_error_t doca_sync_event_start (doca_sync_event *event)

Start a Sync Event to be operate as stand-alone DOCA Core object only.

Parameters

event
Pointer to se event instance to be started.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.

Description

Starting a Sync Event with doca_sync_event_start means it can't be operate as (and converted to) DOCA Context.

doca_error_t doca_sync_event_stop (doca_sync_event *event)

Stop a Sync Event which has been previously started with 'doca_sync_event_start'.

Parameters

event
Pointer to se event instance to be stoped.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.

doca_error_t doca_sync_event_subscriber_add_location_cpu (doca_sync_event *event, doca_dev *dev)

Parameters

event
Target doca_sync_event instance to set.
dev
doca_dev instance associated with CPU.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

Description

Associate a CPU device context as the doca_sync_event Subscriber,

doca_error_t doca_sync_event_subscriber_add_location_dpa (doca_sync_event *event, doca_dpa *dpa)

Associate a DOCA DPA context as the Sync Event Sublisher.

Parameters

event
Target doca_sync_event instance to set.

dpa
doca_dpa instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_subscriber_add_location_dpu (doca_sync_event *event)

Declare Sync Event Publisher as the DPU.

Parameters

event
Target doca_sync_event instance to set.

Returns

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

- DOCA_ERROR_INVALID_VALUE - event argument is a NULL pointer.

doca_error_t doca_sync_event_subscriber_add_location_gpu (doca_sync_event *event, doca_gpu *gpu)

Associate a DOCA GPU context as the Sync Event Subscriber.

Parameters

event
Target doca_sync_event instance to set.

gpu
doca_gpu instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_update_add (doca_sync_event *event, uint64_t value, uint64_t *fetched)

Atomically increase the value of a Sync Event by some value synchronously.

Parameters

event
Target doca_sync_event instance to increment.

value
The value to increment the doca_sync_event value by.

fetched
The value of the doca_sync_event before the operation.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_update_set (doca_sync_event *event, uint64_t value)

Set the value of a Sync Event to some value synchronously.

Parameters

event
Target doca_sync_event instance to set its value.

value
The value to set the doca_sync_event to.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_wait_gt (doca_sync_event *event, uint64_t value, uint64_t mask)

Wait for the value of a Sync Event to reach some value synchronously in a polling busy wait manner.

Parameters

event
Target doca_sync_event instance to wait on.

value
The value to wait for the doca_sync_event to be greater than.

mask
Mask to apply (bitwise AND) on the doca_sync_event value for comparison with wait threshold.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

doca_error_t doca_sync_event_wait_gt_yield (doca_sync_event *event, uint64_t value, uint64_t mask)

Wait for the value of a Sync Event to reach some value synchronously in a periodically busy wait manner.

Parameters

event
Target doca_sync_event instance to wait on.

value
The value to wait for the doca_sync_event to be greater than.

mask
Mask to apply (bitwise AND) on the doca_sync_event value for comparison with wait threshold.

Returns

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

- DOCA_ERROR_INVALID_VALUE - any of the arguments is a NULL pointer.

Description

After each polling iteration, call sched_yield sched_yield() causes the calling thread to relinquish the CPU. The thread is moved to the end of the queue for its static priority and a new thread gets to run.

2.4.13. DOCA Types

Core

DOCA Types introduces types that are common for many libraries.

struct doca_pci_bdf

The PCI address of a device - same as the address in lspci.

enum doca_access_flags

Specifies the permission level for DOCA buffer.

Can be used with doca_mmap_set_permissions() to set permission level. A few notes: DOCA_ACCESS_DPU_READ_ONLY and DOCA_ACCESS_DPU_READ_WRITE are mutually exclusive Buffer can always be read locally by local device, regardless of set permissions local device - doca_dev running in the same process of the mmap remote device - doca_dev running on a different process on a remote machine DPU device - doca_dev running on a process on the DPU OS. This is only relevant when local process is running

on HOST. In case local process is running on DPU the doca_dev is considered a local device.

Values

DOCA_ACCESS_LOCAL_READ_ONLY = 0
DOCA_ACCESS_LOCAL_READ_WRITE = (1<<0)
DOCA_ACCESS_RDMA_READ = (1<<1)
DOCA_ACCESS_RDMA_WRITE = (1<<2)
DOCA_ACCESS_RDMA_ATOMIC = (1<<3)
DOCA_ACCESS_DPU_READ_ONLY = (1<<4)
DOCA_ACCESS_DPU_READ_WRITE = (1<<5)
Allows reading buffer by a DPU device but no write Only relevant for HOST - see doca_mmap_export_dpu()

enum doca_eth_wait_on_time

Type of wait on time the network card can support.

Values

DOCA_ETH_WAIT_ON_TIME_NONE = 0
DOCA_ETH_WAIT_ON_TIME_NATIVE = 1
DOCA_ETH_WAIT_ON_TIME_DPDK = 2

enum doca_gpu_mem_type

Type of memory the GPUNetIO library can allocate.

Values

DOCA_GPU_MEM_GPU = 0
DOCA_GPU_MEM_GPU_CPU = 1
DOCA_GPU_MEM_CPU = 2
DOCA_GPU_MEM_CPU_GPU = 3

enum doca_pci_func_type

Specifies the PCI function type for DOCA representor device.

Values

DOCA_PCI_FUNC_PF = 0
DOCA_PCI_FUNC_VF
DOCA_PCI_FUNC_SF

typedef uint16_t doca_be16_t

Declare DOCA endianity types.

2.5. Comm Channel

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

enum doca_comm_channel_msg_flags

Flags for send/receive functions.

Values

DOCA_CC_MSG_FLAG_NONE = 0

typedef HANDLE doca_event_channel_t

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

< Windows

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

Client side Connect.

Parameters

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

name
identifies the service. Use doca_comm_channel_get_max_service_name_len() to get the maximal service name length.

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

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if no ep object, name or peer_address pointer given. DOCA_ERROR_DRIVER if failed to query the capabilities of the device that was set for the ep or acquire device attributes. DOCA_ERROR_NOT_SUPPORTED if tried to call connect on a device that doesn't have the capability to connect to Comm Channel. DOCA_ERROR_NOT_PERMITTED if the endpoint is already connected. DOCA_ERROR_BAD_STATE if no doca_dev was set. DOCA_ERROR_NO_MEMORY if memory allocation failed. DOCA_ERROR_INITIALIZATION

if initialization of ep connection failed. DOCA_ERROR_CONNECTION_ABORTED if connection failed for any reason (connections rejected or failed).

Description

This function available only for client-side use. As part of the connection process, the client initiates an internal handshake protocol with the service.

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

doca_error_t doca_comm_channel_ep_create (doca_comm_channel_ep_t **ep)

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

Parameters

ep
handle to the newly created endpoint.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if no ep pointer or no attribute object was given. DOCA_ERROR_NO_MEMORY if memory allocation failed during ep creation. DOCA_ERROR_INITIALIZATION if initialization of ep failed. DOCA_ERROR_DRIVER if acquiring device attributes failed.

doca_error_t doca_comm_channel_ep_destroy (doca_comm_channel_ep_t *local_ep)

Release endpoint handle.

Parameters

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

Returns

DOCA_SUCCESS on success. DOCA_ERROR_NOT_CONNECTED if ep does not exist.

Description

The function close the event_channel and release all internal resources. The doca_comm_channel_ep_disconnect() is included as part of the destroy process.

doca_error_t doca_comm_channel_ep_disconnect (doca_comm_channel_ep_t *local_ep, doca_comm_channel_addr_t *peer_addr)

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

Parameters

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

peer_addr
peer address to be disconnect from.

Returns

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

doca_error_t doca_comm_channel_ep_event_handle_arm_recv (doca_comm_channel_ep_t *local_ep)

Arm the event_channel handle for received messages. This function arms the receive completion queue, facilitating blocking on the receive event channel. Blocking should be implemented by the user (poll in Linux, GetQueuedCompletionStatus in Windows).

Parameters

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

Returns

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep object was given.

doca_error_t doca_comm_channel_ep_event_handle_arm_send (doca_comm_channel_ep_t *local_ep)

Arm the event_channel handle for transmitted messages. This function arms the transmit completion queue, facilitating blocking on the transmit event channel. Blocking

should be implemented by the user (poll in Linux, GetQueuedCompletionStatus in Windows).

Parameters

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

Returns

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep object was given.

doca_error_t doca_comm_channel_ep_get_device (doca_comm_channel_ep_t *ep, doca_dev **device)

get device property of endpoint.

Parameters

ep
endpoint from which the property should be retrieved.
device
current device used in endpoint.

Returns

DOCA_SUCCESS if property was returned successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given.

doca_error_t doca_comm_channel_ep_get_device_rep (doca_comm_channel_ep_t *ep, doca_dev_rep **device_rep)

get device representor property of endpoint.

Parameters

ep
endpoint from which the property should be retrieved.
device_rep
current device representor used in endpoint.

Returns

DOCA_SUCCESS if property returned successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given.

doca_error_t doca_comm_channel_ep_get_event_channel (doca_comm_channel_ep_t *local_ep, doca_event_channel_t *send_event_channel, doca_event_channel_t *recv_event_channel)

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

Parameters

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

send_event_channel
handle for send event channel.

recv_event_channel
handle for receive event channel.

Returns

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep was provided or if both event channel output params are null. DOCA_ERROR_BAD_STATE if called before calling doca_comm_channel_ep_listen() or doca_comm_channel_ep_connect(). DOCA_ERROR_NOT_FOUND if another error occurred.

doca_error_t doca_comm_channel_ep_get_max_msg_size (doca_comm_channel_ep_t *ep, uint16_t *max_msg_size)

get maximal msg size property of endpoint. The size returned is the actual size being used and might differ from the size set with doca_comm_channel_ep_set_max_msg_size(), as there is a minimal size requirement. If maximal msg size was not set, using doca_comm_channel_ep_set_max_msg_size(), a default value is used and can be inquired by calling doca_comm_channel_ep_get_max_msg_size().

Parameters

ep
endpoint from which the property should be retrieved.

max_msg_size
maximal msg size used by the endpoint.

Returns

DOCA_SUCCESS if property was returned successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given.

doca_error_t doca_comm_channel_ep_get_recv_queue_size (doca_comm_channel_ep_t *ep, uint16_t *recv_queue_size)

get receive queue size property of endpoint. The size returned is the actual size being used and might differ from the size set with doca_comm_channel_ep_set_recv_queue_size(), as there is a minimal size requirement and the size is rounded up to the closest power of 2. If receive queue size was not set, using doca_comm_channel_ep_set_recv_queue_size(), a default value is used and can be inquired by calling doca_comm_channel_ep_get_recv_queue_size().

Parameters

ep
endpoint from which the property should be retrieved.

recv_queue_size
receive queue size used by the endpoint.

Returns

DOCA_SUCCESS if property was returned successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given.

doca_error_t doca_comm_channel_ep_get_send_queue_size (doca_comm_channel_ep_t *ep, uint16_t *send_queue_size)

get send queue size property of endpoint. The size returned is the actual size being used and might differ from the size set with doca_comm_channel_ep_set_send_queue_size(), as there is a minimal size requirement and the size is rounded up

to the closest power of 2. If send queue size was not set, using doca_comm_channel_ep_set_send_queue_size(), a default value is used and can be inquired by calling doca_comm_channel_ep_get_send_queue_size().

Parameters

ep
endpoint from which the property should be retrieved.

send_queue_size
send queue size used by the endpoint.

Returns

DOCA_SUCCESS if property was returned successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given.

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

Service side listen on all interfaces.

Parameters

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

name
identifies the service. Use doca_comm_channel_get_max_service_name_len() to get the maximal service name length.

Returns

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep object or no name was given. DOCA_ERROR_DRIVER if failed to query the capabilities of the device that was set for the ep. DOCA_ERROR_NOT_SUPPORTED if tried to call listen on a device that doesn't have the capability to be defiend as the service side for Comm Channel. DOCA_ERROR_BAD_STATE if no doca_dev or no doca_dev_rep was set. DOCA_ERROR_NOT_PERMITTED if the endpoint is already listening. DOCA_ERROR_NO_MEMORY if memory allocation failed. DOCA_ERROR_INITIALIZATION if initialization of service failed. DOCA_ERROR_CONNECTION_ABORTED if registration of service failed. DOCA_ERROR_DRIVER if acquiring device attributes failed.

Description

Endpoint will start listening on given devices. After calling this function the user should call doca_comm_channel_ep_recvfrom() in order to get new peers to communicate with.

This function available only for service side use.

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

Receive message from connected client/service.

Parameters

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

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

len

flags
flag for receive command. currently no flags are supported.

peer_addr
received message source address handle

Returns

DOCA_SUCCESS on successful receive. If a message was received, the value pointed by len will be updated with the number of bytes received. DOCA_ERROR_INVALID_VALUE if any of the parameters is NULL. DOCA_ERROR_NOT_CONNECTED if endpoint is service and listen was not called. DOCA_ERROR_AGAIN if no message was received. when returned, the user can use the endpoint's doca_event_channel_t to get indication for a new arrival message. DOCA_ERROR_CONNECTION_RESET if the message received is from a peer_addr that has error. DOCA_ERROR_INITIALIZATION if initialization of the DCI after a send error failed DOCA_ERROR_UNKNOWN if an unknown error occurred.

Description

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

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

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

Parameters

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

msg
pointer to the message to be sent.

len
length in bytes of msg.

flags
flag for send command. currently no flags are supported.

peer_addr
destination address handle of the send operation.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_NOT_CONNECTED if no peer_address was supplied or no connection was found. DOCA_ERROR_INVALID_VALUE if the supplied len was larger than the msgsize given at ep creation or any of the input variables are null. DOCA_ERROR_AGAIN if the send queue is full. when returned, the user can use the endpoint's doca_event_channel_t to get indication for a new empty slot. DOCA_ERROR_CONNECTION_RESET if the provided peer_addr experienced an error and it needs to be disconnected. DOCA_ERROR_INITIALIZATION if initialization of the DCI after a send error failed DOCA_ERROR_UNKNOWN if an unknown error occurred.

doca_error_t doca_comm_channel_ep_set_device (doca_comm_channel_ep_t *ep, doca_dev *device)

set device property for endpoint.

Parameters

ep
endpoint to set the property for.

device
device to use in endpoint.

Returns

DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t doca_comm_channel_ep_set_device_rep (doca_comm_channel_ep_t *ep, doca_dev_rep *device_rep)

set device representor property for endpoint.

Parameters

ep
endpoint to set the property for.

device_rep
device representor to use in endpoint.

Returns

DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t doca_comm_channel_ep_set_max_msg_size (doca_comm_channel_ep_t *ep, uint16_t max_msg_size)

set maximal msg size property for endpoint. The value max_msg_size may be increased internally, the actual value can be queried using doca_comm_channel_ep_get_max_msg_size().

Parameters

ep
endpoint to set the property for.

max_msg_size
maximal msg size to use in endpoint.

Returns

DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t doca_comm_channel_ep_set_recv_queue_size (doca_comm_channel_ep_t *ep, uint16_t recv_queue_size)

set receive queue size property for endpoint. The value recv_queue_size may be increased internally, the actual value can be queried using doca_comm_channel_ep_get_recv_queue_size().

Parameters

ep
endpoint to set the property for.

recv_queue_size
receive queue size to use in endpoint.

Returns

DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t doca_comm_channel_ep_set_send_queue_size (doca_comm_channel_ep_t *ep, uint16_t send_queue_size)

set send queue size property for endpoint. The value send_queue_size may be increased internally, the actual value can be queried using doca_comm_channel_ep_get_send_queue_size().

Parameters

ep
endpoint to set the property for.

send_queue_size
send queue size to use in endpoint.

Returns

DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t doca_comm_channel_get_max_message_size (doca_devinfo *devinfo, uint32_t *max_message_size)

Get the maximum message size supported by comm_channel.

Parameters

devinfo

devinfo that should be inquired for its maximum message size under comm channel limitations.

max_message_size

the maximum message size supported by comm_channel.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or max_message_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

doca_error_t doca_comm_channel_get_max_recv_queue_size (doca_devinfo *devinfo, uint32_t *max_recv_queue_size)

Get the maximum receive queue size supported by comm_channel.

Parameters

devinfo

devinfo that should be inquired for its maximum receive queue size under comm channel limitations.

max_recv_queue_size

the maximum receive queue size supported by comm_channel.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or max_recv_queue_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

doca_error_t doca_comm_channel_get_max_send_queue_size (doca_devinfo *devinfo, uint32_t *max_send_queue_size)

Get the maximum send queue size supported by comm_channel.

Parameters

devinfo

devinfo that should be inquired for its maximum send queue size under comm channel limitations.

max_send_queue_size

the maximum send queue size supported by comm_channel.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or max_send_queue_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

doca_error_t doca_comm_channel_get_max_service_name_len (uint32_t *max_service_name_len)

Get the comm_channel maximum Service name length.

Parameters

max_service_name_len

The comm_channel max service name length, including the terminating null byte (').

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if max_service_name_len is NULL.

doca_error_t doca_comm_channel_get_service_max_num_connections (doca_devinfo *devinfo, uint32_t *max_num_connections)

Get the maximum number of connections the service can hold.

Parameters

devinfo
devinfo that should be inquired for its maximum number of connections.

max_num_connections
the maximum number of connections the service can hold.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or max_num_connections is NULL. DOCA_ERROR_NOT_SUPPORTED if querying this capability is not supported by the device. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

Description

Note:

This capability should be queried only on the service side.

doca_error_t doca_comm_channel_peer_addr_get_recv_bytes (const doca_comm_channel_addr_t *peer_addr, uint64_t *recv_bytes)

get total bytes received from specific peer address

Parameters

peer_addr
Pointer to peer_addr to query statistics for.

recv_bytes
Will contain the number of received bytes from the given peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description

This function will return the total number of bytes received from a given peer_addr, updated to the last time doca_comm_channel_peer_addr_update_info() was called.

doca_error_t doca_comm_channel_peer_addr_get_recv_messages (const doca_comm_channel_addr_t *peer_addr, uint64_t *recv_messages)

get total messages received from specific peer address

Parameters

peer_addr
Pointer to peer_addr to query statistics for.

recv_messages
Will contain the number of received messages from the given peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description

This function will return the total number of messages received from a given peer_addr, updated to the last time doca_comm_channel_peer_addr_update_info() was called.

doca_error_t doca_comm_channel_peer_addr_get_send_bytes (const doca_comm_channel_addr_t *peer_addr, uint64_t *send_bytes)

get total bytes sent to specific peer address

Parameters

peer_addr
Pointer to peer_addr to query statistics for.

send_bytes
Will contain the number of sent messages to the given peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description

This function will return the total number of bytes sent to a given peer_addr, updated to the last time doca_comm_channel_peer_addr_update_info() was called.

doca_error_t doca_comm_channel_peer_addr_get_send_in_flight_messa (const doca_comm_channel_addr_t *peer_addr, uint64_t *send_in_flight_messages)

get number of messages in transmission to a specific peer address

Parameters

peer_addr
Pointer to peer_addr to query statistics for.

send_in_flight_messages
Will contain the number of sent messages in transmission to the given peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description

This function will return the number of messages still in transmission to a specific peer_addr, updated to the last time doca_comm_channel_peer_addr_update_info() was called. This function can be used to make sure all transmissions are finished before disconnection.

doca_error_t doca_comm_channel_peer_addr_get_send_messages (const doca_comm_channel_addr_t *peer_addr, uint64_t *send_messages)

get total messages sent to specific peer address

Parameters

peer_addr
Pointer to peer_addr to query statistics for.

send_messages
Will contain the number of sent messages to the given peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description

This function will return the total number of messages sent to a given peer_addr, updated to the last time doca_comm_channel_peer_addr_update_info() was called.

doca_error_t doca_comm_channel_peer_addr_get_user_data (doca_comm_channel_addr_t *peer_addr, uint64_t *user_data)

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

Parameters

peer_addr
Pointer to peer_addr to extract user_context from.

user_data
will contain the extracted data.

Returns

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

doca_error_t doca_comm_channel_peer_addr_set_user_data (doca_comm_channel_addr_t *peer_addr, uint64_t user_context)

Save 'user_context' in peer_addr handle.

Parameters

peer_addr
Pointer to peer_addr to set user_context to.

user_context
Data to set for peer_addr.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_address is NULL.

Description

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

doca_error_t doca_comm_channel_peer_addr_update_info (doca_comm_channel_addr_t *peer_addr)

update statistics for given peer_addr

Parameters

peer_addr
Pointer to peer_addr to update statistics in.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_addr is NULL. DOCA_ERROR_CONNECTION_INPROGRESS if connection is not yet established. DOCA_ERROR_CONNECTION_ABORTED if the connection failed.

Description

Should be used before calling to any peer_addr information function to update the saved statistics. This function can also be used to check if connection to a given peer_addr is currently connected. If a connection has failed, it is the user's responsibility to call doca_comm_channel_ep_disconnect() to free the peer_addr resources.

2.6. 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 __declspec(deprecated("Symbol is defined as experimental"), DLL_EXPORT_ATTR)

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

#define DOCA_STRUCT_START uint32_t __doca_api_version

Compatibility Helpers

2.7. DOCA COMPRESS engine

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

struct doca_compress_deflate_job

struct doca_compress_lz4_job

enum doca_compress_job_types

Available jobs for DOCA COMPRESS.

Values

DOCA_COMPRESS_DEFLATE_JOB = DOCA_ACTION_COMPRESS_FIRST+1
DOCA_DECOMPRESS_DEFLATE_JOB
DOCA_DECOMPRESS_LZ4_JOB

__DOCA_EXPERIMENTAL doca_ctx *doca_compress_as_ctx (doca_compress *compress)

Parameters

compress
COMPRESS instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

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

doca_error_t doca_compress_create (doca_compress **compress)

Parameters

compress
Pointer to pointer to be set to point to the created doca_compress instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - compress argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_compress.
- DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Create a DOCA COMPRESS instance.

doca_error_t doca_compress_destroy (doca_compress *compress)

Parameters

compress
Pointer to instance to be destroyed.

Returns

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

- DOCA_ERROR_IN_USE - if unable to gain exclusive access to the compress instance or if one or more work queues are still attached. These must be detached first.

Description

Destroy a DOCA COMPRESS instance.

doca_error_t doca_compress_get_max_buf_size (const doca_devinfo *devinfo, doca_compress_job_types job_type, uint64_t *max_buffer_size)

Parameters

devinfo
The DOCA device information

job_type
doca_compress job type. See enum doca_compress_job_types.

max_buffer_size
The max buffer size for DOCA COMPRESS operation in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_compress job.

Description

Get maximum buffer size for DOCA COMPRESS job.

doca_error_t doca_compress_get_max_list_buf_num_elem (const doca_devinfo *devinfo, uint32_t *max_list_num_elem)

Parameters

devinfo
The DOCA device information.

max_list_num_elem
The maximum supported number of elements in DOCA linked-list buffer. The value 1 indicates that only a single element is supported.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximum supported number of elements in DOCA linked-list buffer for compress job.

doca_error_t doca_compress_job_get_supported (doca_devinfo *devinfo, doca_compress_job_types job_type)

Parameters

devinfo
The DOCA device information

job_type
doca_compress job type. See enum doca_compress_job_types.

Returns

DOCA_SUCCESS - in case the job is supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_compress job.

Description

Check if given device is capable for given doca_compress job.

2.8. Environment Configurations

#define DOCA_COMPAT_HELPERS

declares the support/need for compatibility helper utils

2.9. ct

DOCA HW connection tracking library.

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

struct doca_ct_cfg

doca ct global configuration

struct doca_flow_action_desc

action description

struct doca_flow_action_descs

action descriptions

struct doca_flow_action_descs_meta

Metadata action description per field.

struct doca_flow_action_field

extended modification action

struct doca_flow_actions

doca flow actions information

struct doca_flow_aged_query

aged flow query callback context

struct doca_flow_cfg

doca flow global configuration

struct doca_flow_encap_action

doca flow encap data information

struct doca_flow_fwd

forwarding configuration

struct doca_flow_header_format

doca flow packet format

struct doca_flow_match

doca flow matcher information

struct doca_flow_meta

doca flow meta data

struct doca_flow_mirror_target

doca flow mirror target

struct doca_flow_monitor

doca monitor action configuration

struct doca_flow_ordered_list

struct doca_flow_pipe_attr

pipe attributes

struct doca_flow_pipe_cfg

pipeline configuration

struct doca_flow_port_cfg

doca flow port configuration

struct doca_flow_push_action

doca flow push data information

struct doca_flow_query

flow query result

struct doca_flow_resource_crypto_cfg

doca flow crypto resource configuration

struct doca_flow_resource_meter_cfg

doca flow meter resource configuration

struct doca_flow_resource_mirror_cfg

doca flow mirror resource configuration

struct doca_flow_resource_rss_cfg

doca flow rss resource configuration

struct doca_flow_resources

doca flow resource quota

struct doca_flow_shared_resource_cfg

doca flow shared resource configuration

struct doca_flow_shared_resource_result

flow shared resources query result

enum doca_ct_flags

CT flags.

Values

DOCA_CT_FLAG_STATS = 1u<<0
Enable counter for internal pipes
DOCA_CT_FLAG_WORKER_STATS = 1u<<1
Enable worker counter dump
DOCA_CT_FLAG_NO_AGING = 1u<<2
Bypass aging scan

enum doca_ct_session_type

CT l3 session types.

Values

DOCA_CT_SESSION_IPV4
IPv4 session.
DOCA_CT_SESSION_IPV6
IPv6 session.
DOCA_CT_SESSION_BOTH
Total session.
DOCA_CT_SESSION_MAX
Max session types.

enum doca_ct_types

CT flags.

Values

DOCA_CT_TYPE_KNOWN
Known TCP/UDP connection
DOCA_CT_TYPE_SYN

New connection packet

DOCA_CT_TYPE_FIN

Connection finish packet

DOCA_CT_TYPE_RST

Connection reset packet

enum doca_flow_action_type

action type enumeration

Values

DOCA_FLOW_ACTION_AUTO = 0

DOCA_FLOW_ACTION_CONSTANT

DOCA_FLOW_ACTION_SET

DOCA_FLOW_ACTION_ADD

DOCA_FLOW_ACTION_COPY

DOCA_FLOW_ACTION_MAX

enum doca_flow_cfg_flags

doca flow configuration flags

Values

DOCA_FLOW_CFG_PIPE_MISS_MON = (1<<0)

Counter pipe miss flow and query with doca_flow_query_pipe_miss()

enum doca_flow_entry_op

doca flow entry operation

Values

DOCA_FLOW_ENTRY_OP_ADD

Add entry

DOCA_FLOW_ENTRY_OP_DEL

Delete entry

DOCA_FLOW_ENTRY_OP_UPD

Update entry

enum doca_flow_entry_status

doca flow entry status

Values

DOCA_FLOW_ENTRY_STATUS_IN_PROCESS

DOCA_FLOW_ENTRY_STATUS_SUCCESS

DOCA_FLOW_ENTRY_STATUS_ERROR

enum doca_flow_flags_type

doca flow flags type

Values

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

enum doca_flow_fwd_type

forwarding action type

Values

DOCA_FLOW_FWD_NONE = 0
No forward action be set
DOCA_FLOW_FWD_RSS
Forwards packets to rss
DOCA_FLOW_FWD_PORT
Forwards packets to one port
DOCA_FLOW_FWD_PIPE
Forwards packets to another pipe
DOCA_FLOW_FWD_DROP
Drops packets
DOCA_FLOW_FWD_TARGET
Forwards packets to target
DOCA_FLOW_FWD_ORDERED_LIST_PIPE
Forwards packet to a specific entry in an ordered list pipe.

enum doca_flow_l2_valid_header

doca flow l2 valid headers

Values

DOCA_FLOW_L2_VALID_HEADER_VLAN_0 = (1<<0)
first vlan
DOCA_FLOW_L2_VALID_HEADER_VLAN_1 = (1<<1)
second vlan

enum doca_flow_match_tcp_flags

doca flow match flags

Values

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

enum doca_flow_ordered_list_element_type

Type of an ordered list element.

Values

DOCA_FLOW_ORDERED_LIST_ELEMENT_ACTIONS
Ordered list element is struct doca_flow_actions, the next element is struct doca_flow_action_descs associated with the current element.
DOCA_FLOW_ORDERED_LIST_ELEMENT_ACTION_DESCS
Ordered list element is struct doca_flow_action_descs. If the previous element type is ACTIONS, the current element is associated with it. Otherwise the current element is ordered w.r.t. the previous one.
DOCA_FLOW_ORDERED_LIST_ELEMENT_MONITOR
Ordered list element is struct doca_flow_monitor.

enum doca_flow_pipe_domain

doca flow pipe domain

Values

DOCA_FLOW_PIPE_DOMAIN_DEFAULT = 0
Default pipe domain for actions on ingress traffic
DOCA_FLOW_PIPE_DOMAIN_SECURE_INGRESS
Pipe domain for secure actions on ingress traffic
DOCA_FLOW_PIPE_DOMAIN_EGRESS

Pipe domain for actions on egress traffic

DOCA_FLOW_PIPE_DOMAIN_SECURE_EGRESS

Pipe domain for actions on egress traffic

enum doca_flow_pipe_type

doca flow pipe type

Values

DOCA_FLOW_PIPE_BASIC

Flow pipe

DOCA_FLOW_PIPE_CONTROL

Control pipe

DOCA_FLOW_PIPE_LPM

longest prefix match (LPM) pipe

DOCA_FLOW_PIPE_CT

Connection Tracking pipe

DOCA_FLOW_PIPE_ACL

ACL pipe

DOCA_FLOW_PIPE_ORDERED_LIST

Ordered list pipe

DOCA_FLOW_PIPE_HASH

Hash pipe

enum doca_flow_port_type

doca flow port type

Values

DOCA_FLOW_PORT_DPDK_BY_ID

dpdk port by mapping id

enum doca_flow_push_action_type

doca flow push action type

Values

DOCA_FLOW_PUSH_ACTION_VLAN

enum doca_flow_shared_resource_type

Shared resource supported types.

Values

DOCA_FLOW_SHARED_RESOURCE_METER

Shared meter type

DOCA_FLOW_SHARED_RESOURCE_COUNT

Shared counter type

DOCA_FLOW_SHARED_RESOURCE_RSS

Shared rss type

DOCA_FLOW_SHARED_RESOURCE_CRYPTO

Shared crypto action type

DOCA_FLOW_SHARED_RESOURCE_MIRROR

Shared mirror type

DOCA_FLOW_SHARED_RESOURCE_MAX

Shared max supported types

enum doca_flow_target_type

doca flow target type

Values

DOCA_FLOW_TARGET_KERNEL

enum doca_rss_type

rss offload types

Values

DOCA_FLOW_RSS_IPV4 = (1<<0)

rss by ipv4 head

DOCA_FLOW_RSS_IPV6 = (1<<1)

rss by ipv6 head

DOCA_FLOW_RSS_UDP = (1<<2)

rss by udp head

DOCA_FLOW_RSS_TCP = (1<<3)

rss by tcp head

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

doca flow entry process callback

typedef (*doca_flow_shared_resource_unbind_cb) (enum doca_flow_shared_resource_type, uint32_t shared_resource_id, void* bindable_obj)

doca flow shared resource unbind callback

__DOCA_EXPERIMENTAL void doca_ct_destroy (void)

Destroy the doca ct.

Description

Release all the resources used by doca ct.

Must be invoked before doca flow detroy.

doca_error_t doca_ct_init (const doca_ct_cfg *cfg)

Initialize the doca ct.

Parameters

cfg
CT configuration.

Returns

0 on success, a negative errno value otherwise.

Description

This is the global initialization function for doca ct. 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 ct initialization and global configurations.

Must be invoked after Doca Flow initilization, before port start.

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

Handle aging of flows in queue.

Parameters

port
Port to handle aging

queue
Queue identifier.

quota
Max time quota in micro seconds for this function to handle aging.

entries
User input entries array for the aged flows.

len
User input length of entries array.

Returns

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

Description

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

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

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

Process entries in queue.

Parameters

port
Port

pipe_queue
Queue identifier.

timeout
Max time in micro seconds for this function to process entries. Process once if timeout is 0

max_processed_entries
Flow entries number to process If it is 0, it will proceed until timeout.

Returns

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

- DOCA_ERROR_DRIVER - driver error.

Description

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

doca_error_t doca_flow_get_target (doca_flow_target_type type, doca_flow_target **target)

Get doca flow forward target.

Parameters

type
Target type.

target
Target handler on success

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - unsupported type.

doca_error_t doca_flow_init (const doca_flow_cfg *cfg)

Initialize the doca flow.

Parameters

cfg
Port configuration, see doca_flow_cfg for details.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_NOT_SUPPORTED - unsupported configuration.
- DOCA_ERROR_UNKNOWN - otherwise.

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_error_t doca_flow_mpls_label_decode (const doca_flow_header_mpls *mpls, uint32_t *label, uint8_t *traffic_class, uint8_t *ttl, bool *bottom_of_stack)

Decode an MPLS label header.

Parameters

mpls
Pointer to MPLS structure to decode.
label
Pointer to fill MPLS label value.
traffic_class
Pointer to fill MPLS traffic class value.

ttl
Pointer to fill MPLS TTL value.
bottom_of_stack
Pointer to fill whether this MPLS is bottom of stack.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Note:

: All output variables are in cpu-endian.

doca_error_t doca_flow_mpls_label_encode (uint32_t label, uint8_t traffic_class, uint8_t ttl, bool bottom_of_stack, doca_flow_header_mpls *mpls)

Prepare an MPLS label header in big-endian.

Parameters

label
The label value - 20 bits.
traffic_class
Traffic class - 3 bits.
ttl
Time to live - 8 bits
bottom_of_stack
Whether this MPLS is bottom of stack.
mpls
Pointer to MPLS structure to fill.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Note:

: All input variables are in cpu-endian.

doca_error_t doca_flow_pipe_acl_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_match *match_mask, const uint32_t priority, const doca_flow_fwd *fwd, const doca_flow_flags_type flag, void *usr_ctx, doca_flow_pipe_entry **entry)

Add one new entry to a acl pipe.

Parameters

pipe_queue
Queue identifier.

pipe
Pointer to pipe.

match
Pointer to match, indicate specific packet match information.

match_mask
Pointer to match mask information.

priority
Priority value

fwd
Pointer to fwd actions.

flag
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

usr_ctx
Pointer to user context.

entry
The entry inserted.

Returns

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

Description

This API will populate the acl entries

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

Add one new entry to a pipe.

Parameters

pipe_queue
Queue identifier.

pipe
Pointer to pipe.

match
Pointer to match, indicate specific packet match information.

actions
Pointer to modify actions, indicate specific modify information.

monitor
Pointer to monitor actions.

fwd
Pointer to fwd actions.

flags
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

usr_ctx
Pointer to user context.

entry
Pipe entry handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

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_error_t doca_flow_pipe_control_add_entry (uint16_t pipe_queue, uint32_t priority, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_actions *actions, const doca_flow_action_descs *action_descs, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, doca_flow_pipe_entry **entry)

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.

actions
Pointer to modify actions, indicate specific modify information.

action_descs
action descriptions

monitor
Pointer to monitor actions.
fwd
Pointer to fwd actions.
entry
Pipe entry handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

Description

Refer to doca_flow_pipe_add_entry.

doca_error_t doca_flow_pipe_create (const doca_flow_pipe_cfg *cfg, const doca_flow_fwd *fwd, const doca_flow_fwd *fwd_miss, doca_flow_pipe **pipe)

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.
pipe
Pipe handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - unsupported pipe type.
- DOCA_ERROR_DRIVER - driver error.

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_pipe_destroy (doca_flow_pipe *pipe)

Destroy one pipe.

Parameters

pipe
Pointer to pipe.

Description

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

__DOCA_EXPERIMENTAL void doca_flow_pipe_dump (doca_flow_pipe *pipe, FILE *f)

Dump pipe information.

Parameters

pipe
Pointer to doca flow pipe.

f
The output file of the pipe information.

doca_flow_entry_status doca_flow_pipe_entry_get_status (doca_flow_pipe_entry *entry)

Get entry's status.

Parameters

entry
pipe entry

Returns

entry's status

doca_error_t doca_flow_pipe_hash_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, uint32_t entry_index, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, const doca_flow_flags_type flags, void *usr_ctx, doca_flow_pipe_entry **entry)

Add one new entry to an hash pipe.

Parameters

pipe_queue
Queue identifier.
pipe
Pointer to pipe.
entry_index
Static index in pipe for this entry.
actions
Pointer to modify actions, indicate specific modify information.
monitor
Pointer to monitor actions.
fwd
Pointer to forward actions.
flags
Flow entry will be pushed to HW immediately or not. enum doca_flow_flags_type.
usr_ctx
Pointer to user context.

entry
Pipe entry handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

Description

Refer to doca_flow_pipe_add_entry.

doca_error_t doca_flow_pipe_lpm_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, const doca_flow_flags_type flag, void *usr_ctx, doca_flow_pipe_entry **entry)

Add one new entry to a lpm pipe.

Parameters

pipe_queue
Queue identifier.
pipe
Pointer to pipe.
match
Pointer to match, indicate specific packet match information.
match_mask
Pointer to match mask information.
actions
Pointer to modify actions, indicate specific modify information.
monitor
Pointer to monitor actions.
fwd
Pointer to fwd actions.
flag
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

usr_ctx
Pointer to user context.
entry
Pipe entry handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

Description

This API will populate the lpm entries

doca_error_t doca_flow_pipe_lpm_update_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, const doca_flow_flags_type flags, doca_flow_pipe_entry *entry)

Update the lpm pipe entry with new actions.

Parameters

pipe_queue
Queue identifier.
pipe
Pointer to pipe.
actions
Pointer to modify actions, indicate specific modify information.
monitor
Pointer to monitor actions.
fwd
Pointer to fwd actions.
flags
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.
entry
The pipe entry to be updated.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

doca_error_t doca_flow_pipe_ordered_list_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, uint32_t idx, const doca_flow_ordered_list *ordered_list, const doca_flow_fwd *fwd, doca_flow_flags_type flags, void *user_ctx, doca_flow_pipe_entry **entry)

Parameters

pipe_queue
Queue identifier.

pipe
Pipe handle.

idx
Unique entry index. It is the user's responsibility to ensure uniqueness.

ordered_list
Ordered list with pointers to struct doca_flow_actions and struct doca_flow_monitor at the same indices as they were at the pipe creation time. If the configuration contained an element of struct doca_flow_action_descs, the corresponding array element is ignored and can be NULL.

fwd
Entry forward configuration.

flags
Entry insertion flags.

user_ctx
Opaque context for the completion callback.

entry
The entry inserted.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.

- DOCA_ERROR_DRIVER - driver error.

Description

Add an entry to the ordered list pipe.

doca_error_t doca_flow_pipe_rm_entry (uint16_t pipe_queue, uint32_t flags, void *usr_ctx, doca_flow_pipe_entry *entry)

Free one pipe entry.

Parameters

pipe_queue
Queue identifier.

flags
Flow entry will be removed from hw immediately or not. enum doca_flow_flags_type.

usr_ctx
The pointer to user context.

entry
The pipe entry to be removed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - unsupported pipe type.
- DOCA_ERROR_UNKNOWN - otherwise.

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_error_t doca_flow_pipe_update_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, const doca_flow_flags_type flags, doca_flow_pipe_entry *entry)

Update the pipe entry with new actions.

Parameters

pipe_queue
Queue identifier.

pipe
Pointer to pipe.

actions
Pointer to modify actions, indicate specific modify information.

monitor
Pointer to monitor actions.

fwd
Pointer to fwd actions.

flags
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

entry
The pipe entry to be updated.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - driver error.

doca_error_t doca_flow_port_pair (doca_flow_port *port, doca_flow_port *pair_port)

pair two doca flow ports.

Parameters

port
Pointer to doca flow port.

pair_port
Pointer to the pair port.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

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

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

__DOCA_EXPERIMENTAL void doca_flow_port_pipes_dump (doca_flow_port *port, FILE *f)

Dump pipes of one port.

Parameters

port
Pointer to doca flow port.

f
The output file of the pipe information.

Description

Dump all pipes information belong to this port.

__DOCA_EXPERIMENTAL void doca_flow_port_pipes_flush (doca_flow_port *port)

Flush pipes of one port.

Parameters

port
Pointer to doca flow port.

Description

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

__DOCA_EXPERIMENTAL uint8_t *doca_flow_port_priv_data (doca_flow_port *port)

Get pointer of user private data.

Parameters

port
Port struct.

Returns

Private data head pointer.

Description

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

doca_error_t doca_flow_port_start (const doca_flow_port_cfg *cfg, doca_flow_port **port)

Start a doca port.

Parameters

cfg
Port configuration, see doca_flow_cfg for details.
port
Port handler on success.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_NOT_SUPPORTED - unsupported port type.
- DOCA_ERROR_UNKNOWN - otherwise.

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_error_t doca_flow_port_stop (doca_flow_port *port)

Stop a doca port.

Parameters

port
Port struct.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

Stop the port, disable the traffic, destroy the doca port, free all resources of the port.

__DOCA_EXPERIMENTAL doca_flow_port *doca_flow_port_switch_get (void)

Get doca flow switch port.

Description

The application could use this function to get the doca switch port, then create pipes and pipe entries on this port.

doca_error_t doca_flow_query_entry (doca_flow_pipe_entry *entry, doca_flow_query *query_stats)

Extract information about specific entry.

Parameters

entry
The pipe entry to query.

query_stats
Data retrieved by the query.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

Query the packet statistics about specific pipe entry

doca_error_t doca_flow_query_pipe_miss (doca_flow_pipe *pipe, doca_flow_query *query_stats)

Extract information about pipe miss entry.

Parameters

pipe
The pipe to query.

query_stats
Data retrieved by the query.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

Query the packet statistics about specific pipe miss entry

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

Configure a single shared resource.

Parameters

type
Shared resource type.

id
Shared resource id.

cfg
Pointer to a shared resource configuration.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_NOT_SUPPORTED - unsupported shared resource type.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

This API can be used by bounded and unbounded resources.

doca_error_t doca_flow_shared_resources_bind (doca_flow_shared_resource_type type, uint32_t *res_array, uint32_t res_array_len, void *bindable_obj)

Binds a bulk of shared resources to a bindable object.

Parameters

type
Shared resource type.

res_array
Array of shared resource IDs.
res_array_len
Shared resource IDs array length.
bindable_obj
Pointer to an allowed bindable object, use NULL to bind globally.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_NOT_SUPPORTED - unsupported shared resource type.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

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

doca_error_t doca_flow_shared_resources_query (doca_flow_shared_resource_type type, uint32_t *res_array, doca_flow_shared_resource_result *query_results_array, uint32_t array_len)

Extract information about shared counter.

Parameters

type
Shared object type.
res_array
Array of shared objects IDs to query.
query_results_array
Data array retrieved by the query.
array_len
Number of objects and their query results in their arrays (same number).

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.

- DOCA_ERROR_NOT_SUPPORTED - unsupported shared resource type.
- DOCA_ERROR_UNKNOWN - otherwise.

Description

Query an array of shared objects of a specific type.

#define DOCA_FLOW_META_EXT 16

External meta data size in bytes.

#define DOCA_FLOW_META_MAX 20

Max meta data size in bytes.

#define DOCA_FLOW_SWITCH doca_flow_port_switch_get()

Mapping to doca flow switch port.

#define DOCA_FLOW_VLAN_MAX 2

Max number of vlan headers.

2.10. DOCA DMA engine

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

struct doca_dma_job_memcpy

struct doca_dma_memcpy_result

enum doca_dma_devinfo_caps

Possible DMA device capabilities.

Values

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

enum doca_dma_job_types

Available jobs for DMA.

Values

DOCA_DMA_JOB_MEMCPY = DOCA_ACTION_DMA_FIRST+1

__DOCA_EXPERIMENTAL doca_ctx *doca_dma_as_ctx (doca_dma *dma)

Parameters

dma

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

Returns

Non NULL upon success, NULL otherwise.

Description

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

doca_error_t doca_dma_create (doca_dma **dma)

Parameters

dma

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - dma argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_dma.
- DOCA_ERROR_INITIALIZATION - failed to initialise a mutex.

Description

Create a DOCA DMA instance.

doca_error_t doca_dma_destroy (doca_dma *dma)

Parameters

dma
Pointer to instance to be destroyed.

Returns

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

- DOCA_ERROR_IN_USE - Unable to gain exclusive access to the dma instance.
- DOCA_ERROR_IN_USE - One or more work queues are still attached. These must be detached first.

doca_error_t doca_dma_get_max_buf_size (const doca_devinfo *devinfo, uint64_t *buf_size)

Parameters

devinfo
The DOCA device information.
buf_size
The maximum supported buffer size in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximum supported buffer size for DMA job.

doca_error_t doca_dma_get_max_list_buf_num_elem (const doca_devinfo *devinfo, uint32_t *max_list_num_elem)

Parameters

devinfo
The DOCA device information.

max_list_num_elem
The maximum supported number of elements in a given DOCA linked-list buffer, such that 1 indicates no linked-list buffer support.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximum supported number of elements in a given DOCA linked-list buffer for DMA job.

doca_error_t doca_dma_job_get_supported (doca_devinfo *devinfo, doca_dma_job_types job_type)

Parameters

devinfo
The DOCA device information

job_type
DMA job_type available through this device. see enum doca_dma_job_types.

Returns

DOCA_SUCCESS - in case device supports job_type. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this DMA job.

Description

Check if given device is capable of excuting a specific DMA job.

2.13. Deep packet inspection

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

struct doca_dpi_job

DOCA_DPI job definition.

struct doca_dpi_parsing_info

L2-L4 flow information, used to uniquely define a flow.

struct doca_dpi_result

DOCA_DPI result definition.

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_flow_status_t

Status of enqueued flow.

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_job_types

DOCA_DPI job types.

Values

DOCA_DPI_JOB = DOCA_ACTION_DPI_FIRST+1

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 doca_ctx *doca_dpi_as_ctx (doca_dpi *dpi)

Convert DOCA_DPI instance into context for use with workQ.

Parameters

dpi
DOCA_DPI instance. This must remain valid until after the context is no longer required.

Returns

Non NULL - doca_ctx object on success. Error:

- NULL.

doca_error_t doca_dpi_create (doca_dpi **dpi)

Create a DOCA_DPI instance.

Parameters

dpi

Instance pointer to be created, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - not enough memory for allocation.
- DOCA_ERROR_NOT_SUPPORTED - the required engine is not supported.

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), when we still use the dpdk-regex as backend.

doca_error_t doca_dpi_destroy (doca_dpi *dpi)

Destroy DOCA_DPI instance.

Parameters

dpi

Instance to be destroyed, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the attached workq is not detached yet. Please call doca_ctx_stop().

doca_error_t doca_dpi_flow_create (doca_dpi *dpi, doca_workq *workq, const doca_dpi_parsing_info *parsing_info, doca_dpi_flow_ctx **flow_ctx)

Creates a new flow for a workq.

Parameters

dpi
The DOCA_DPI instance.

workq
The DPI workq on which to create the flow

parsing_info
L3/L4 information.

flow_ctx
The created flow.

Description

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

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - memory allocation failure.

doca_error_t doca_dpi_flow_destroy (doca_dpi_flow_ctx *flow_ctx)

Destroys a flow on a workq.

Parameters

flow_ctx
The flow context to destroy.

Description

Should be called when a flow is terminated or times out

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_dpi_get_flow_match (const doca_dpi_flow_ctx *flow_ctx, doca_dpi_result *result)

Get the latest match of a flow.

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

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_get_max_sig_match_len (const doca_dpi *dpi, uint16_t *max_sig_match_len)

Get the maximum signature match length of a DPI instance.

Parameters

dpi
The DOCA_DPI instance.

max_sig_match_len
Output of the maximum signature match length

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_get_per_workq_max_flows (const doca_dpi *dpi, uint32_t *per_workq_max_flows)

Return the configures maximum active flows per workq.

Parameters

dpi
The DOCA_DPI instance.

per_workq_max_flows
The maximum parallel flows supported per workq.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_get_per_workq_packet_pool_size (const doca_dpi *dpi, uint32_t *per_workq_packet_pool_size)

Return the value in the corresponding set command.

Parameters

dpi
The DOCA_DPI instance.

per_workq_packet_pool_size
Output of the maximum inflight job number.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_get_signature (const doca_dpi *dpi, uint32_t sig_id, doca_dpi_sig_data *sig_data)

Get a specific signature data of a DPI instance.

Parameters

dpi
The DOCA_DPI instance.

sig_id
The signature ID.

sig_data
Output of the sig data.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_get_signatures (const doca_dpi *dpi, doca_dpi_sig_data **sig_data, uint32_t *total_sigs)

Get all signatures of a DPI instance.

Parameters

dpi
The DOCA_DPI instance.

sig_data
Output of the sig data.

total_sigs
Output of the number of signatures.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

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_error_t doca_dpi_get_stats (const doca_dpi *dpi, bool clear, doca_dpi_stat_info *stats)

Get DPI statistics.

Parameters

dpi
The DOCA_DPI instance.

clear
Clear the statistics after fetching them.

stats
Output struct containing the statistics.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

doca_error_t doca_dpi_is_supported (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Device can be used with doca_dpi. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - Device cannot be used with doca_dpi.

Description

Determine if a given device is suitable for use with doca_dpi.

doca_error_t doca_dpi_job_get_supported (const doca_devinfo *devinfo, doca_dpi_job_types job_type)

Check if given device is capable of excuting a specific DOCA_DPI job.

Parameters

devinfo
The DOCA device information

job_type
DOCA_DPI job_type to check for support.

Returns

DOCA_SUCCESS - in case device supports job_type. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this DOCA_DPI job.

doca_error_t doca_dpi_set_in_order_mode (const doca_dpi *dpi, bool enabled)

Enable or disable 'in order' mode.

Parameters

dpi
The DOCA_DPI instance.

enabled
Set to true to turn 'in order' mode on, false to disable it.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

Description

Due to the parallel nature of pattern matching accelerators that may be used by DPI, there is no guarantee that results from jobs sent will be returned in the same order.

Enabling 'in order' mode guarantees that results returned per workq will be in the order they were sent.

Note:

This mode is disabled by default. Choosing to enable it may have a negative impact on performance.

doca_error_t doca_dpi_set_max_sig_match_len (doca_dpi *dpi, uint16_t max_sig_match_len)

Set the maximum length that DPI guarantees to provide a match on.

Parameters

dpi

DOCA_DPI Instance, MUST NOT BE NULL.

max_sig_match_len

The value to be set.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

Description

It includes across consecutive packets. Must be <= 5000 For example: Signature = A.*B max_sig_match_len = 5 DPI guarantee that ACCCB will be found (len <= 5) DPI does not guarantee that ACCCCCCCB will be found (len > 5)

doca_error_t doca_dpi_set_per_workq_max_flows (doca_dpi *dpi, uint32_t per_workq_max_flows)

Set the maximum number of active flows that can be supported by a DOCA_DPI workq.

Parameters

dpi

DOCA_DPI Instance, MUST NOT BE NULL.

per_workq_max_flows

The value to be set.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

doca_error_t doca_dpi_set_per_workq_packet_pool_size (doca_dpi *dpi, uint32_t per_workq_packet_pool_size)

Set the maximum number of packets packets that can be stored for a DOCA_DPI workq.

Parameters

dpi
DOCA_DPI Instance, MUST NOT BE NULL.

per_workq_packet_pool_size
The value to be set.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

Description

DOCA_DPI may require that packets are stored to ensure signatures are detected that occur accross multiple packets of the same flow. This value sets the maximum number of packets that can be simultaenously stored for all flows beings processed on a workq. After this function is called, all the "memory" of the previous packets enqueued are deleted.

doca_error_t doca_dpi_set_signatures (doca_dpi *dpi, const char *cdo_file)

Set the cdo file.

Parameters

dpi
DOCA_DPI Instance, MUST NOT BE NULL.

cdo_file
CDO file created by the DPI compiler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

Description

The cdo file contains signature information. The cdo file must be loaded before doca_dpi instance is started. So the typical usage sequence is: create doca_dpi instance --> set signatures --> start doca_dpi instance.

Database update: When a new signatures database is available, the doca_dpi instance must be stopped first, then, the user may call this function again. So the typical usage sequence is: stop doca_dpi instance --> set new signatures. The newly loaded CDO must contain the signatures of the previously loaded CDO or result will be undefined.

2.15. DOCA ETH RXQ

DOCA ETH RXQ library.

enum doca_eth_rxq_type

RX queue type.

Values

DOCA_ETH_RXQ_TYPE_CYCLIC = 0

__DOCA_EXPERIMENTAL doca_ctx *doca_eth_rxq_as_doca_ctx (doca_eth_rxq *eth_rxq)

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

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

Returns

Non NULL upon success, NULL otherwise.

doca_error_t doca_eth_rxq_create (doca_eth_rxq **eth_rxq)

Create a DOCA ETH RXQ instance.

Parameters

eth_rxq
Pointer to pointer to be set to point to the created doca_eth_rxq instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - eth_rxq argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to allocate resources.
- DOCA_ERROR_INITIALIZATION - failed to initialize eth_rxq.

doca_error_t doca_eth_rxq_destroy (doca_eth_rxq *eth_rxq)

Destroy a DOCA ETH RXQ instance.

Parameters

eth_rxq
Pointer to instance to be destroyed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - eth_rxq argument is a NULL pointer.

- DOCA_ERROR_BAD_STATE - the associated ctx was not stopped before calling doca_eth_rxq_destroy().

doca_error_t doca_eth_rxq_get_flow_queue_id (doca_eth_rxq *eth_rxq, uint16_t *flow_queue_id)

Get the DPDK queue ID of the doca_eth receive queue. can only be called after calling doca_ctx_start().

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

flow_queue_id
The queue ID to be used in rte_flow or doca_flow

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context was not started.

doca_error_t doca_eth_rxq_get_gpu_handle (const doca_eth_rxq *eth_rxq, doca_gpu_eth_rxq **eth_rxq_ext)

Get a gpu handle of a doca_eth_rxq.

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

eth_rxq_ext
A doca gpu eth_rxq handle.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is not started.

Description

This method should be used after ctx is started. The expected flow is as follows: 1. bind the ctx to a gpu device using doca_ctx_set_data_path_on_gpu() 2. start the ctx using doca_ctx_start() 3. call doca_eth_rxq_get_gpu_handle() to get the gpu_handle

doca_error_t doca_eth_rxq_get_max_packet_size_supported (const doca_devinfo *devinfo, uint16_t *max_packet_size)

Get the maximum packet size supported by the device.

Parameters

devinfo
Pointer to doca_devinfo instance.

max_packet_size
The max packet size.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.

doca_error_t doca_eth_rxq_get_pkt_buffer_size (const doca_eth_rxq *eth_rxq, uint32_t *size)

Get the required size for the Eth packet buffer of a doca_eth_rxq.

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

size
The required size for the eth packet buffer.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - in case of uninitialized queue size (packet_size, num_packets).

Description

This function should be used for calculating the minimum size of the doca_mmap given to doca_eth_rxq_set_pkt_buffer().

> Note:
>
> Must be called after doca_eth_rxq_set_num_packets() and doca_eth_rxq_set_max_packet_size().

doca_error_t doca_eth_rxq_get_type_supported (const doca_devinfo *devinfo, doca_eth_rxq_type type, uint8_t *type_supported)

Check if RX queue type is supported.

Parameters

devinfo
Pointer to doca_devinfo instance.

type
RX queue type - see enum doca_eth_rxq_type

type_supported
Flag to indicate if type is supported.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.

doca_error_t doca_eth_rxq_set_max_packet_size (doca_eth_rxq *eth_rxq, uint16_t max_pkt_sz)

Set max packet size property for doca_eth_rxq. can only be called before calling doca_ctx_start().

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

max_pkt_sz
Max packet size to use in context.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_rxq_set_num_packets (doca_eth_rxq *eth_rxq, uint32_t num_packets)

Set queue size property for doca_eth_rxq. can only be called before calling doca_ctx_start().

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

num_packets
Queue max number of packets to use in context.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_rxq_set_pkt_buffer (doca_eth_rxq *eth_rxq, doca_mmap *mmap, uint32_t offset, uint32_t size)

Set Eth packet buffer for a doca_eth_rxq. can only be called before calling doca_ctx_start().

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

mmap
The mmap consist of the memrange for the Eth packet buffer.

offset
The offset from mmap start to set the packet buffer.

size
The size of the Eth packet buffer.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_rxq_set_type (doca_eth_rxq *eth_rxq, doca_eth_rxq_type type)

Set RX queue type property for doca_eth_rxq. can only be called before calling doca_ctx_start().

Parameters

eth_rxq
Pointer to doca_eth_rxq instance.

type
RX queue type - see enum doca_eth_rxq_type

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

Description

Note:

in doca_eth_rxq_create() the default type is DOCA_ETH_RXQ_TYPE_CYCLIC

2.16. DOCA ETH TXQ

DOCA ETH TXQ library.

enum doca_eth_txq_type

TX queue type.

Values

DOCA_ETH_TXQ_TYPE_CYCLIC = 0

__DOCA_EXPERIMENTAL doca_ctx *doca_eth_txq_as_doca_ctx (doca_eth_txq *eth_txq)

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

Parameters

eth_txq
Pointer to doca_eth_txq instance.

Returns

Non NULL upon success, NULL otherwise.

doca_error_t doca_eth_txq_calculate_timestamp (doca_eth_txq *eth_txq, uint64_t timestamp_ns, uint64_t *wait_on_time_value)

Calculate timestamp to use when setting the wait on time on the Tx queue.

Parameters

eth_txq
Pointer to doca_eth_txq instance.
timestamp_ns
Timestamp to indicate when send packets
wait_on_time_value
Value to use to enqueue wait on time in send queue

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_DRIVER - error query underlying network card driver
- DOCA_ERROR_NOT_PERMITTED - wait on time clock is not enabled on the network card.

doca_error_t doca_eth_txq_create (doca_eth_txq **eth_txq)

Create a DOCA ETH TXQ instance.

Parameters

eth_txq
Pointer to pointer to be set to point to the created doca_eth_txq instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - eth_txq argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_eth_txq.
- DOCA_ERROR_INITIALIZATION - failed to initialize eth_txq.

doca_error_t doca_eth_txq_destroy (doca_eth_txq *eth_txq)

Destroy a DOCA ETH TXQ instance.

Parameters

eth_txq
Pointer to instance to be destroyed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - eth_txq argument is a NULL pointer.
- DOCA_ERROR_BAD_STATE - the associated ctx was not stopped before calling doca_eth_txq_destroy().

doca_error_t doca_eth_txq_get_chksum_offload_supported (const doca_devinfo *devinfo, uint8_t *offload_supported)

Check if checksum offload is supported by the device.

Parameters

devinfo
Pointer to doca_devinfo instance.

offload_supported
Flag to indicate if checksum offload is supported.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.

doca_error_t doca_eth_txq_get_gpu_handle (const doca_eth_txq *eth_txq, doca_gpu_eth_txq **eth_txq_ext)

Get a gpu handle of a doca_eth_txq.

Parameters

eth_txq
Pointer to doca_eth_txq instance.

eth_txq_ext
A doca gpu eth_txq handle.

Returns

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

Description

This method should be used after ctx is started. The expected flow is as follows: 1. bind the ctx to a gpu device using doca_ctx_set_data_path_on_gpu() 2. start the ctx using doca_ctx_start() 3. call doca_eth_txq_get_gpu_handle() to get the gpu_handle

doca_error_t doca_eth_txq_get_max_queue_size_supported (const doca_devinfo *devinfo, uint32_t *max_queue_size)

Get the maximum queue size supported by the device.

Parameters

devinfo
Pointer to doca_devinfo instance.

max_queue_size
The max queue size.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.

doca_error_t doca_eth_txq_get_type_supported (const doca_devinfo *devinfo, doca_eth_txq_type type, uint8_t *type_supported)

Check if TX queue type is supported.

Parameters

devinfo
Pointer to doca_devinfo instance.

type
TX queue type - see enum doca_eth_txq_type

type_supported
Flag to indicate if type is supported.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.

doca_error_t doca_eth_txq_get_wait_on_time_offload_supported (const doca_dev *dev, doca_eth_wait_on_time * *wait_on_time_mode)

Check if wait on time offload is supported by the network device.

Parameters

dev
Pointer to doca_dev instance.

wait_on_time_mode
Offload wait on time mode (native or DPDK)

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_DRIVER - error query underlying network card driver
- DOCA_ERROR_NOT_SUPPORTED - real-time clock is not enable on the network card.

doca_error_t doca_eth_txq_set_l3_chksum_offload (doca_eth_txq *eth_txq)

Set offload for the calculation of IPv4 checksum (L3) on transmitted packets. can only be called before calling doca_ctx_start().

Parameters

eth_txq
Pointer to doca_eth_txq instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_txq_set_l4_chksum_offload (doca_eth_txq *eth_txq)

Set offload for the calculation of TCP/UDP checksum (L4) on transmitted packets. can only be called before calling doca_ctx_start().

Parameters

eth_txq
Pointer to doca_eth_txq instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_txq_set_queue_size (doca_eth_txq *eth_txq, uint32_t size)

Set queue size property for doca_eth_txq. can only be called before calling doca_ctx_start().

Parameters

eth_txq
Pointer to doca_eth_txq instance.
size
Queue size to use in context.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_eth_txq_set_type (doca_eth_txq *eth_txq, doca_eth_txq_type type)

Set TX queue type property for doca_eth_txq. can only be called before calling doca_ctx_start().

Parameters

eth_txq
Pointer to doca_eth_txq instance.

type
TX queue type - see enum doca_eth_txq_type

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

Description

Note:

in doca_eth_txq_create() the default type is DOCA_ETH_TXQ_TYPE_CYCLIC

doca_error_t doca_eth_txq_set_wait_on_time_offload (doca_eth_txq *eth_txq)

Set offload to enable wait on time feature on the queue.

Parameters

eth_txq
Pointer to doca_eth_txq instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.
- DOCA_ERROR_NOT_PERMITTED - Wait on time HW support but network device clock is not in REAL TIME mode.

2.17. flow net define

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

enum doca_flow_crypto_action_type

doca flow crypto operation action type

Values

DOCA_FLOW_CRYPTO_ACTION_NONE = 0
No crypto action performed
DOCA_FLOW_CRYPTO_ACTION_ENCRYPT
Perform encryption
DOCA_FLOW_CRYPTO_ACTION_DECRYPT
Perform decryption/authentication

enum doca_flow_crypto_header_type

doca flow crypto operation encapsulation header type

Values

DOCA_FLOW_CRYPTO_HEADER_NONE = 0
No network header involved
DOCA_FLOW_CRYPTO_HEADER_IPV4
IPv4 network header type
DOCA_FLOW_CRYPTO_HEADER_IPV6
IPv6 network header type
DOCA_FLOW_CRYPTO_HEADER_IPV4_UDP
IPv4 + UDP network header type
DOCA_FLOW_CRYPTO_HEADER_IPV6_UDP
IPv6 + UDP network header type

enum doca_flow_crypto_icv_size

doca flow crypto protocol Integrity Check Value size

Values

DOCA_FLOW_CRYPTO_ICV_DEFAULT = 0
Use default ICV size for specified protocol.
DOCA_FLOW_CRYPTO_ICV_8B = 8
ICV size is 8B
DOCA_FLOW_CRYPTO_ICV_12B = 12
ICV size is 12B
DOCA_FLOW_CRYPTO_ICV_16B = 16
ICV size is 16B

enum doca_flow_crypto_net_type

doca flow crypto operation network mode type

Values

DOCA_FLOW_CRYPTO_NET_NONE = 0

No network header involved

DOCA_FLOW_CRYPTO_NET_TUNNEL
Tunnel network header

DOCA_FLOW_CRYPTO_NET_TRANSPORT
Tramsport network header

enum doca_flow_crypto_protocol_type

doca flow crypto operation protocol type

Values

DOCA_FLOW_CRYPTO_PROTOCOL_NONE = 0
No security protocol engaged

DOCA_FLOW_CRYPTO_PROTOCOL_NISP
NISP protocol action

DOCA_FLOW_CRYPTO_PROTOCOL_ESP
IPsec ESP protocol action

enum doca_flow_crypto_reformat_type

doca flow crypto operation reformat type

Values

DOCA_FLOW_CRYPTO_REFORMAT_NONE = 0
No reformat action performed

DOCA_FLOW_CRYPTO_REFORMAT_ENCAP
Perform encapsulation action

DOCA_FLOW_CRYPTO_REFORMAT_DECAP
Perform decapsulation action

2.18. Flow

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

struct doca_flow_grpc_bindable_obj

bindable object configuration

struct doca_flow_grpc_fwd

forwarding configuration wrapper

struct doca_flow_grpc_pipe_cfg

pipeline configuration wrapper

enum doca_flow_grpc_bindable_obj_type

doca flow grpc bindable object types

Values

DOCA_FLOW_GRPC_BIND_TYPE_PIPE
bind resource to a pipe

DOCA_FLOW_GRPC_BIND_TYPE_PORT
bind resource to a port

DOCA_FLOW_GRPC_BIND_TYPE_NULL
bind resource globally

doca_error_t doca_flow_grpc_aging_handle (uint16_t port_id, uint16_t queue, uint64_t quota, uint64_t *entries_id, int len, int *nb_aged_flow)

RPC call for doca_flow_aging_handle().

Parameters

port_id
Port id to handle aging

queue
Queue identifier.

quota
Max time quota in micro seconds for this function to handle aging.

entries_id
User input entries array for the aged flows.

len
User input length of entries array.

nb_aged_flow
Number of aged flow.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

__DOCA_EXPERIMENTAL void doca_flow_grpc_client_create (const char *grpc_address)

Initialize a channel to DOCA flow grpc server.

Parameters

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

Description

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

doca_error_t doca_flow_grpc_destroy (void)

RPC call for doca_flow_destroy().

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise

doca_error_t doca_flow_grpc_entries_process (uint16_t port_id, uint16_t pipe_queue, uint64_t timeout, uint32_t max_processed_entries, int *num_processed_entries)

RPC call for doca_flow_grpc_entries_process().

Parameters

port_id
Port ID
pipe_queue
Queue identifier.
timeout
Max time in micro seconds for this function to process entries. Process once if timeout is 0

max_processed_entries
Flow entries number to process If it is 0, it will proceed until timeout.
num_processed_entries
Number of entries processed

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_init (const doca_flow_cfg *cfg)

RPC call for doca_flow_init().

Parameters

cfg
Program configuration, see doca_flow_cfg for details.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

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

RPC call for doca_flow_pipe_add_entry().

Parameters

pipe_queue
Queue identifier.
pipe_id
Pipe ID.
match
Pointer to match, indicate specific packet match information.
actions
Pointer to modify actions, indicate specific modify information.
monitor
Pointer to monitor actions.

client_fwd
Pointer to fwd actions.
flags
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.
entry_id
Created entry ID on success.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_pipe_control_add_entry (uint16_t pipe_queue, uint8_t priority, uint64_t pipe_id, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_grpc_fwd *client_fwd, uint64_t *entry_id)

RPC call for doca_flow_pipe_control_add_entry().

Parameters

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

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_pipe_create (const doca_flow_grpc_pipe_cfg *cfg, const doca_flow_grpc_fwd *fwd, const doca_flow_grpc_fwd *fwd_miss, uint64_t *pipe_id)

RPC call for doca_flow_pipe_create().

Parameters

cfg
Pipe configuration, see doca_flow_grpc_pipe_cfg for details.

fwd
Fwd configuration for the pipe.

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

pipe_id
Created pipe ID on success.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_pipe_destroy (uint64_t pipe_id)

RPC call for doca_flow_pipe_destroy().

Parameters

pipe_id
Pipe ID.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise

doca_error_t doca_flow_grpc_pipe_dump (uint64_t pipe_id, FILE *f)

RPC call for doca_flow_pipe_dump().

Parameters

pipe_id
pipe ID.

f
The output file of the pipe information.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise

doca_error_t doca_flow_grpc_pipe_entry_get_status (uint64_t entry_id, doca_flow_entry_status * *entry_status)

RPC call for doca_flow_pipe_entry_get_status().

Parameters

entry_id
pipe entry ID
entry_status
entry's status

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_pipe_lpm_add_entry (uint16_t pipe_queue, uint64_t pipe_id, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_grpc_fwd *client_fwd, const doca_flow_flags_type flag, uint64_t *entry_id)

RPC call for doca_flow_pipe_lpm_add_entry().

Parameters

pipe_queue
Queue identifier.
pipe_id
Pipe ID.
match
Pointer to match, indicate specific packet match information.

match_mask
Pointer to match mask information.
actions
Pointer to modify actions, indicate specific modify information.
monitor
Pointer to monitor actions.
client_fwd
Pointer to fwd actions.
flag
Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.
entry_id
Created entry ID on success.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_pipe_rm_entry (uint16_t pipe_queue, uint64_t entry_id, uint32_t flags)

RPC call for doca_flow_grpc_pipe_rm_entry().

Parameters

pipe_queue
Queue identifier.
entry_id
The entry ID to be removed.
flags
Flow entry will removed from hw immediately or not. enum doca_flow_flags_type.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_port_pair (uint16_t port_id, uint16_t pair_port_id)

RPC call for doca_flow_port_pair().

Parameters

port_id
port ID.

pair_port_id
pair port ID.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_port_pipes_dump (uint16_t port_id, FILE *f)

RPC call for doca_flow_port_pipes_dump().

Parameters

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

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise

doca_error_t doca_flow_grpc_port_pipes_flush (uint16_t port_id)

RPC call for doca_flow_port_pipes_flush().

Parameters

port_id
Port ID.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise

doca_error_t doca_flow_grpc_port_start (const doca_flow_port_cfg *cfg, uint16_t *port_id)

RPC call for doca_flow_port_start().

Parameters

cfg
Port configuration, see doca_flow_port_cfg for details.
port_id
Created port ID on success.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_port_stop (uint16_t port_id)

RPC call for doca_flow_port_stop().

Parameters

port_id
Port ID.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_port_switch_get (uint16_t *port_id)

RPC call for doca_flow_port_switch_get().

Parameters

port_id
Switch port ID

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_query_entry (uint64_t entry_id, doca_flow_query *query_stats)

RPC call for doca_flow_query_entry().

Parameters

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

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_query_pipe_miss (uint64_t pipe_id, doca_flow_query *query_stats)

RPC call for doca_flow_query_pipe_miss().

Parameters

pipe_id
The pipe ID to query.

query_stats
Data retrieved by the query.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

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

RPC call for doca_flow_shared_resource_cfg().

Parameters

type
Shared resource type.

id
Shared resource id.

cfg
Pointer to a shared resource configuration.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

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

RPC call for doca_flow_shared_resources_bind().

Parameters

type
Shared resource type.

res_array
Array of shared resource IDs.

res_array_len
Shared resource IDs array length.

bindable_obj_id
Pointer to a bindable object ID.

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

doca_error_t doca_flow_grpc_shared_resources_query (doca_flow_shared_resource_type type, uint32_t *res_array, doca_flow_shared_resource_result *query_results_array, uint32_t array_len)

RPC call for doca_flow_shared_resources_query().

Parameters

type
Shared object type.

res_array
Array of shared objects IDs to query.

query_results_array
Data array retrieved by the query.

array_len
Number of objects and their query results in their arrays (same number).

Returns

DOCA_SUCCESS on success and DOCA_ERROR otherwise.

2.19. 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_header_eth

doca flow eth header

struct doca_flow_header_eth_vlan

doca flow vlan header

struct doca_flow_header_geneve

doca flow GENEVE header.

struct doca_flow_header_icmp

doca flow icmp header in match data

struct doca_flow_header_ip4

doca flow ipv4 header in match data

struct doca_flow_header_ip6

doca flow ipv6 header in match data

struct doca_flow_header_l4_port

doca flow tcp or udp port header in match data

struct doca_flow_header_mpls

doca flow MPLS header.

struct doca_flow_header_tcp

doca flow tcp header in match data

struct doca_flow_header_udp

doca flow udp header in match data

struct doca_flow_ip_addr

doca flow ip address

struct doca_flow_tun

doca flow tunnel information

enum doca_flow_l3_type

doca flow layer 3 packet type

Values

DOCA_FLOW_L3_TYPE_NONE = 0
l3 type is not set
DOCA_FLOW_L3_TYPE_IP4
l3 type is ipv4
DOCA_FLOW_L3_TYPE_IP6
l3 type is ipv6

enum doca_flow_l4_type_ext

doca flow layer 4 packet extend type

Values

DOCA_FLOW_L4_TYPE_EXT_NONE = 0
l4 ext type is not set
DOCA_FLOW_L4_TYPE_EXT_TCP
l4 ext type is tcp
DOCA_FLOW_L4_TYPE_EXT_UDP
l4 ext type is udp
DOCA_FLOW_L4_TYPE_EXT_ICMP
l4 ext type is icmp
DOCA_FLOW_L4_TYPE_EXT_ICMP6
l4 ext type is icmp6

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
DOCA_FLOW_TUN_NISP
tunnel is nisp type
DOCA_FLOW_TUN_AUDP

tunnel is audp type

DOCA_FLOW_TUN_ESP
tunnel is ipsec esp type

DOCA_FLOW_TUN_MPLS_O_UDP
tunnel is mpls over udp type

DOCA_FLOW_TUN_GENEVE
tunnel is geneve type

#define DOCA_ETHER_ADDR_LEN (6)

length of ether add length.

#define DOCA_ETHER_TYPE_IPV4 (0x0800)

Ethernet frame types IPv4 Protocol.

#define DOCA_ETHER_TYPE_IPV6 (0x86DD)

IPv6 Protocol.

#define DOCA_ETHER_TYPE_TEB (0x6558)

Transparent Ethernet Bridging.

#define DOCA_FLOW_AUDP_DWORD 6

AUDP header maximal length in dwords

#define DOCA_FLOW_AUDP_HEADER_LEN (DOCA_FLOW_AUDP_DWORD * sizeof(doca_be32_t))

AUDP header maximal length in bytes

#define DOCA_FLOW_CRYPTO_KEY_LEN_MAX 32

Crypto key maximal length in bytes

#define DOCA_FLOW_CRYPTO_REFORMAT_LEN_MAX (DOCA_ETHER_ADDR_LEN * 2 + \ sizeof(doca_be16_t) + \ sizeof(doca_be16_t) * 2 * 2 + \ sizeof(doca_be32_t) * 15 + \ sizeof(doca_be32_t) * 2 + \ DOCA_FLOW_NISP_HEADER_LEN)

NISP/ESP tunnel header may consist of:

- Ethernet addresses
- Ethernet type
- optional VLAN and 802.1Q headers
- IPv4 (with full options) or IPv6 (w/o options)
- optional UDP header
- NISP or ESP header

#define DOCA_FLOW_ESP_HEADER_LEN (2 * sizeof(doca_be32_t))

IPsec ESP header maximal length in bytes

#define DOCA_FLOW_GENEVE_DEFAULT_PORT (6081)

default GENEVE port id.

#define DOCA_FLOW_MPLS_DEFAULT_PORT (6635)

default MPLS port id.

#define DOCA_FLOW_MPLS_LABELS_MAX 5

Max MPLS labels in single match.

#define DOCA_FLOW_NISP_DWORD 10

NISP header maximal length in dwords

#define DOCA_FLOW_NISP_HEADER_LEN (DOCA_FLOW_NISP_DWORD * sizeof(doca_be32_t))

NISP header maximal length in bytes

#define DOCA_GTPU_PORT (2152)

gtpu upd port id.

#define DOCA_NISP_DEFAULT_PORT (1000)

default nisp/audp port id.

#define DOCA_PROTO_GRE (47)

Cisco GRE tunnels (rfc 1701,1702).

#define DOCA_PROTO_ICMP (1)

Internet Control Message Protocol v4.

#define DOCA_PROTO_ICMP6 (58)

Internet Control Message Protocol v6.

#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.

2.20. DOCA GPUNETIO

DOCA GPUNETIO library.

enum doca_gpu_semaphore_status

Semaphore list of possible statuses.

Values

DOCA_GPU_SEMAPHORE_STATUS_FREE = 0
DOCA_GPU_SEMAPHORE_STATUS_READY = 1
DOCA_GPU_SEMAPHORE_STATUS_DONE = 2
DOCA_GPU_SEMAPHORE_STATUS_HOLD = 3
DOCA_GPU_SEMAPHORE_STATUS_ERROR = 4
DOCA_GPU_SEMAPHORE_STATUS_EXIT = 5

doca_error_t doca_gpu_create (const char *gpu_bus_id, doca_gpu **gpu_dev)

Create a DOCA GPUNETIO handler.

Parameters

gpu_bus_id
GPU PCIe address.

gpu_dev
Pointer to the newly created gpu device handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - gpu_dev argument is a NULL pointer.
- DOCA_ERROR_NOT_FOUND - GPU not found at the input PCIe address
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_gpu.

doca_error_t doca_gpu_destroy (doca_gpu *gpu_dev)

Destroy a DOCA GPUNETIO handler.

Parameters

gpu_dev
Pointer to handler to be destroyed.

Returns

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

doca_error_t doca_gpu_mem_alloc (doca_gpu *gpu_dev, size_t size, size_t alignment, doca_gpu_mem_type mtype, void **memptr_gpu, void **memptr_cpu)

Parameters

gpu_dev
DOCA GPUNetIO handler.
size
Buffer size in bytes.
alignment
Buffer memory alignment. If 0, the return is a pointer that is suitably aligned for any kind of variable (in the same manner as malloc()). Otherwise, the return is a pointer that is a multiple of *align*. Alignment value must be a power of two.
mtype
Type of memory buffer. See enum doca_gpu_memtype for reference.
memptr_gpu
GPU memory pointer. Must be used with CUDA API and within CUDA kernels.
memptr_cpu
CPU memory pointer. Must be used for CPU direct access to the memory.

Returns

Non NULL memptr_gpu pointer on success, NULL otherwise. Non NULL memptr_cpu pointer on success in case of DOCA_GPU_MEM_CPU_GPU and DOCA_GPU_MEM_GPU_CPU, NULL otherwise. DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

- DOCA_ERROR_NO_MEMORY - if an error occurred dealing with GPU memory.

Description

Allocate a GPU accessible memory buffer. Assumes DPDK has been already attached with doca_gpu_to_dpdk(). According to the memory type specified, the buffer can be allocated in:

- DOCA_GPU_MEM_GPU memptr_gpu is not NULL while memptr_cpu is NULL.
- DOCA_GPU_MEM_GPU_CPU both memptr_gpu and memptr_cpu are not NULL.
- DOCA_GPU_MEM_CPU_GPU both memptr_gpu and memptr_cpu are not NULL.
- DOCA_GPU_MEM_CPU is not supported. Use regular malloc instead.

doca_error_t doca_gpu_mem_free (doca_gpu *gpu, void *memptr_gpu)

Parameters

gpu
DOCA GPUNetIO handler.

memptr_gpu
GPU memory pointer to be freed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Free a GPU memory buffer. Only memory allocated with doca_gpu_mem_alloc() can be freed with this function.

doca_error_t doca_gpu_semaphore_create (doca_gpu *gpu_dev, doca_gpu_semaphore **semaphore)

Parameters

gpu_dev
DOCA GPUNetIO handler.

semaphore
Pointer to the newly created semaphore handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Create a DOCA GPUNetIO semaphore.

doca_error_t doca_gpu_semaphore_destroy (doca_gpu_semaphore *semaphore)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Stop a DOCA GPUNetIO semaphore.

doca_error_t doca_gpu_semaphore_get_custom_info_addr (doca_gpu_semaphore *semaphore_cpu, uint32_t idx, void **custom_info)

Parameters

semaphore_cpu
DOCA GPUNetIO semaphore CPU handler.
idx
DOCA GPUNetIO semaphore item index.
custom_info
DOCA GPUNetIO semaphore custom info pointer.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Get pointer to DOCA GPUNetIO semaphore item associated custom info. This can be done only if custom info memory was set to DOCA_GPU_MEM_GPU_CPU or DOCA_GPU_MEM_CPU_GPU.

doca_error_t doca_gpu_semaphore_get_gpu_handle (doca_gpu_semaphore *semaphore_cpu, doca_gpu_semaphore_gpu **semaphore_gpu)

Parameters

semaphore_cpu
DOCA GPUNetIO semaphore CPU handler.

semaphore_gpu
DOCA GPUNetIO semaphore GPU handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Get a DOCA GPUNetIO semaphore handle for GPU. This can be used within a CUDA kernel.

doca_error_t doca_gpu_semaphore_get_status (doca_gpu_semaphore *semaphore_cpu, uint32_t idx, doca_gpu_semaphore_status *status)

Parameters

semaphore_cpu
DOCA GPUNetIO semaphore CPU handler.

idx
DOCA GPUNetIO semaphore item index.

status
DOCA GPUNetIO semaphore status.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Get DOCA GPUNetIO semaphore status from CPU. This can be done only if item memory was set to DOCA_GPU_MEM_GPU_CPU or DOCA_GPU_MEM_CPU_GPU.

doca_error_t doca_gpu_semaphore_set_custom_info (doca_gpu_semaphore *semaphore, uint32_t nbytes, doca_gpu_mem_type mtype)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

nbytes
DOCA GPUNetIO semaphore number of items.

mtype
DOCA GPUNetIO semaphore memory type.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Attach to each item in the DOCA GPUNetIO semaphore a custom user-defined structure defined through a number of bytes reserved for each item. Specify also which memory to use for the custom info items.

doca_error_t doca_gpu_semaphore_set_items_num (doca_gpu_semaphore *semaphore, uint32_t num_items)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

num_items
DOCA GPUNetIO semaphore number of items.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Allocate DOCA GPUNetIO semaphore number of items.

doca_error_t doca_gpu_semaphore_set_memory_type (doca_gpu_semaphore *semaphore, doca_gpu_mem_type mtype)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

mtype
DOCA GPUNetIO semaphore items type of memory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Type of memory to be used to allocate DOCA GPUNetIO semaphore items. It determines semaphore visibility: GPU only or GPU and CPU.

doca_error_t doca_gpu_semaphore_set_status (doca_gpu_semaphore *semaphore_cpu, uint32_t idx, doca_gpu_semaphore_status status)

Parameters

semaphore_cpu
DOCA GPUNetIO semaphore CPU handler.

idx
DOCA GPUNetIO semaphore item index.

status
DOCA GPUNetIO semaphore status.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Set DOCA GPUNetIO semaphore status from CPU. This can be done only if item memory was set to DOCA_GPU_MEM_GPU_CPU or DOCA_GPU_MEM_CPU_GPU.

doca_error_t doca_gpu_semaphore_start (doca_gpu_semaphore *semaphore)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Start a DOCA GPUNetIO semaphore. Attributes can't be set after this point.

doca_error_t doca_gpu_semaphore_stop (doca_gpu_semaphore *semaphore)

Parameters

semaphore
DOCA GPUNetIO semaphore handler.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

Stop a DOCA GPUNetIO semaphore.

#define DOCA_GPUNETIO_VOLATILE (*(volatile typeof(x) *)&(x))

Macro to temporarily cast a variable to volatile.

2.21. IPsec

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

struct doca_encryption_key

IPSec encryption key.

struct doca_ipsec_sa_attr_egress

IPSec sa egress attributes - attributes for outgoing data.

struct doca_ipsec_sa_attr_ingress

IPSec sa egress attributes - attributes for incoming data.

struct doca_ipsec_sa_attr_sn

IPSec sa sn attributes - attributes for sequence number - only if SN or AR enabled.

struct doca_ipsec_sa_attrs

IPSec attributes for create jobs.

struct doca_ipsec_sa_create_job

DOCA IPSec SA creation job.

struct doca_ipsec_sa_destroy_job

DOCA IPSec SA destroy job.

struct doca_ipsec_sa_event_attrs

IPSec sa events attributes - when turned on will trigger an event.

enum doca_encryption_key_type

IPSec encryption key type.

Values

DOCA_ENCRYPTION_KEY_AESGCM_128
size of 128 bit

DOCA_ENCRYPTION_KEY_AESGCM_256
size of 256 bit

enum doca_ipsec_direction

IPSec direction of the key, incoming packets or outgoing.

Values

DOCA_IPSEC_DIRECTION_INGRESS_DECRYPT = 0

incoming packets, decription

DOCA_IPSEC_DIRECTION_EGRESS_ENCRYPT = 1

outgoing packets, encription

enum doca_ipsec_icv_length

IPSec icv length.

Values

DOCA_IPSEC_ICV_LENGTH_8 = 8

size of 8 bit

DOCA_IPSEC_ICV_LENGTH_12 = 12

size of 12 bit

DOCA_IPSEC_ICV_LENGTH_16 = 16

size of 16 bit

enum doca_ipsec_job_types

Doca ipsec action type enums, used to specify ipsec job types.

Values

DOCA_IPSEC_JOB_SA_CREATE = DOCA_ACTION_IPSEC_FIRST+1

create sa object

DOCA_IPSEC_JOB_SA_DESTROY

destroy sa object

enum doca_ipsec_replay_win_size

IPSec replay window size.

Values

DOCA_IPSEC_REPLAY_WIN_SIZE_32 = 32

size of 32 bit

DOCA_IPSEC_REPLAY_WIN_SIZE_64 = 64

size of 64 bit

DOCA_IPSEC_REPLAY_WIN_SIZE_128 = 128

size of 128 bit

DOCA_IPSEC_REPLAY_WIN_SIZE_256 = 256

size of 256 bit

doca_error_t doca_ipsec_antireplay_get_supported (const doca_devinfo *devinfo)

Get is device support antireplay_enable capabilities.

Parameters

devinfo
The DOCA device information

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given capabilitie.

__DOCA_EXPERIMENTAL doca_ctx *doca_ipsec_as_ctx (doca_ipsec *ctx)

Convert IPSec instance into context for use with workQ.

Parameters

ctx
IPSEC instance. This must remain valid until after the context is no longer required.

Returns

Non NULL - doca_ctx object on success. Error:

- NULL.

doca_error_t doca_ipsec_create (doca_ipsec **ctx)

Create a DOCA IPSEC instance.

doca_error_t doca_ipsec_destroy (doca_ipsec *ctx)

Destroy DOCA IPSEC instance.

Parameters

ctx
Instance to be destroyed, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the ctx still in use by one or more workQs.

doca_error_t doca_ipsec_job_get_supported (doca_devinfo *devinfo, doca_ipsec_job_types job_type)

Check if given device is capable for given doca_ipsec job.

Parameters

devinfo
The DOCA device information

job_type
doca_ipsec job type. See enum doca_ipsec_job_types.

Returns

DOCA_SUCCESS - in case the job is supported. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_ipsec job.

__DOCA_EXPERIMENTAL doca_ipsec_sa *doca_ipsec_sa_from_result (doca_event *ev)

Convert IPSec event job into sa object.

Parameters

ev
event of ipsec job

Returns

Non NULL - sa object of ipsec. Error:

- NULL.

doca_error_t doca_ipsec_sequence_number_get_supported (const doca_devinfo *devinfo)

Get is device support sn_enabled capabilities.

Parameters

devinfo
The DOCA device information

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given capabilitie.

doca_error_t doca_ipsec_set_sa_pool_size (doca_ipsec *ctx, uint32_t pool_size)

set the sa pool size for sa objects that are return by doca_ipsec_sa_create_job

Parameters

ctx
IPSEC instance.
pool_size
Number of items to have available. default is 20,000

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of ipsec instance. DOCA_ERROR_IN_USE - ipsec instance is currently started.

Description

Note:

The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context

2.22. Logging Management

Define functions for internal and external logging management

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

doca_log_registrator

Registers log source on program start.
cppClassifierVisibility: visibility=public

enum doca_log_level

log levels

Values

DOCA_LOG_LEVEL_CRIT = 20
Critical log level
DOCA_LOG_LEVEL_ERROR = 30
Error log level
DOCA_LOG_LEVEL_WARNING = 40
Warning log level
DOCA_LOG_LEVEL_INFO = 50
Info log level
DOCA_LOG_LEVEL_DEBUG = 60
Debug log level

typedef (*log_flush_callback) (char* buffer)

logging backend flush() handler

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

Generates a log message.

Parameters

level
Log level enum DOCA_LOG_LEVEL.
source
The log source identifier defined by doca_log_source_register.
line
The line number this log originated from.

format
printf(3) arguments, format and variables.

Returns

DOCA error code.

Description

This should not be used, please prefer using DOCA_LOG...

doca_error_t 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.

Returns

DOCA error code.

Description

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

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

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.

backend
Logging backend that wraps the given buffer (only valid if no error occurred).

Returns

DOCA error code.

Description

Creates a new logging backend. The logger will write each log record at the beginning of this buffer.

doca_error_t doca_log_create_fd_backend (int fd, doca_logger_backend **backend)

Create a logging backend with an fd stream.

Parameters

fd
The file descriptor (int) for the logger's backend.
backend
Logging backend that wraps the given fd (only valid if no error occurred).

Returns

DOCA error code.

Description

Creates a new logging backend.

doca_error_t doca_log_create_file_backend (FILE *fptr, doca_logger_backend **backend)

Create a logging backend with a FILE* stream.

Parameters

fptr
The FILE * for the logger's stream.
backend
Logging backend that wraps the given fptr (only valid if no error occurred).

Returns

DOCA error code.

Description

Creates a new logging backend.

doca_error_t doca_log_create_syslog_backend (const char *name, doca_logger_backend **backend)

Create a logging backend with a syslog output.

Parameters

name
The syslog name for the logger's backend.

backend
Logging backend that exposes the desired syslog functionality (only valid if no error occurred).

Returns

DOCA error code.

Description

Creates a new logging backend.

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

Generates a log message for DLOG operations.

Parameters

level
Log level enum DOCA_LOG_LEVEL.

source
The log source identifier defined by doca_log_source_register.

line
The line number this log originated from.

format
printf(3) arguments, format and variables.

Returns

DOCA error code.

Description

Note:

This function is thread safe.

__DOCA_EXPERIMENTAL uint16_t doca_log_get_bucket_time (void)

Get the timespan of the rate-limit bucket.

Returns

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

__DOCA_EXPERIMENTAL uint16_t doca_log_get_quantity (void)

Get the quantity of the rate-limit bucket.

Returns

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

__DOCA_EXPERIMENTAL uint32_t doca_log_global_level_get (void)

Get the global log level.

Returns

Log level enum DOCA_LOG_LEVEL.

Description

Dynamically query for the global log level, any log under this level will be shown. The global level is used as the initial value when a new logger backend is created.

doca_error_t doca_log_global_level_set (uint32_t level)

Set the log level of ALL logger backends.

Parameters

level
Log level enum DOCA_LOG_LEVEL.

Returns

DOCA error code.

Description

Dynamically change the log level of ALL the logger backends, any log under this level will be shown. Newly created logger backends will use this as their default log level.

doca_error_t doca_log_rate_bucket_register (int source, int *bucket)

Register a new rate bucket.

Parameters

source
The log source identifier defined by doca_log_source_register.
bucket
Bucket identifier that was allocated to this log source (only valid if no error occurred).

Returns

DOCA error code.

Description

Will return the identifier associated with the new bucket.

doca_error_tdoca_error_t __DOCA_EXPERIMENTAL doca_log_rate_limit (uint32_t level, int source, int line, int bucket, const char *format, ...)

Generates a log message with rate limit.

Parameters

level
Log level enum DOCA_LOG_LEVEL.

source
The log source identifier defined by doca_log_source_register.

line
The line number this log originated from.

bucket
The bucket identifier defined by doca_log_rate_bucket_register.

format
printf(3) arguments, format and variables.

Description

This should not be used, please prefer using DOCA_LOG_RATE_LIMIT...

__DOCA_EXPERIMENTAL void doca_log_set_bucket_time (uint16_t bucket_time)

Set the timespan of the rate-limit bucket.

Parameters

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

__DOCA_EXPERIMENTAL void doca_log_set_quantity (uint16_t quantity)

Set the quantity of the rate-limit bucket.

Parameters

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

doca_error_t doca_log_source_destroy (int source)

Destroy a log source.

Parameters

source

The source identifier of source to be destroyed, as allocated by doca_log_source_register.

Returns

DOCA error code.

Description

Destroys a given log source as part of the teardown process of the running program.

Note:

Used automatically via DOCA_LOG_REGISTER, not recommended to call it directly.

doca_error_t doca_log_source_register (const char *source_name, int *source)

Register a log source.

Parameters

source_name

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

source

Source identifier that was allocated to this log source name (only valid if no error occurred).

Returns

DOCA error code.

Description

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

Note:

Recommended to only be used via DOCA_LOG_REGISTER.

#define DOCA_DLOG do { \ } while (0)

Generates a development log message.

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

#define DOCA_DLOG_CRIT DOCA_DLOG(CRIT, format, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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_source, __LINE__, format, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

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, ##__VA_ARGS__)

Generates an INFO log message.

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

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

Generates a log message with rate limit.

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

#define DOCA_LOG_RATE_LIMIT_CRIT DOCA_LOG_RATE_LIMIT(CRIT, format, ##__VA_ARGS__)

Generates a CRITICAL rate limited log message.

#define DOCA_LOG_RATE_LIMIT_DBG DOCA_LOG_RATE_LIMIT(DEBUG, format, ##__VA_ARGS__)

Generates a DEBUG rate limited log message.

#define DOCA_LOG_RATE_LIMIT_ERR DOCA_LOG_RATE_LIMIT(ERROR, format, ##__VA_ARGS__)

Generates an ERROR rate limited log message.

#define DOCA_LOG_RATE_LIMIT_INFO DOCA_LOG_RATE_LIMIT(INFO, format, ##__VA_ARGS__)

Generates an INFO rate limited log message.

#define DOCA_LOG_RATE_LIMIT_WARN DOCA_LOG_RATE_LIMIT(WARNING, format, ##__VA_ARGS__)

Generates a WARNING rate limited log message.

#define DOCA_LOG_WARN DOCA_LOG(WARNING, format, ##__VA_ARGS__)

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.

2.23. DOCA RDMA

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

struct doca_rdma_gid

struct doca_rdma_job_atomic

struct doca_rdma_job_read_write

struct doca_rdma_job_recv

struct doca_rdma_job_send

struct doca_rdma_result

enum doca_rdma_job_types

Available jobs for RDMA.

Values

DOCA_RDMA_JOB_RECV = DOCA_ACTION_RDMA_FIRST+1
DOCA_RDMA_JOB_SEND
DOCA_RDMA_JOB_SEND_IMM
DOCA_RDMA_JOB_READ
DOCA_RDMA_JOB_WRITE
DOCA_RDMA_JOB_WRITE_IMM
DOCA_RDMA_JOB_ATOMIC_CMP_SWP
DOCA_RDMA_JOB_ATOMIC_FETCH_ADD

enum doca_rdma_opcode_t

Job result opcodes

Values

DOCA_RDMA_OPCODE_RECV_SEND = 0
DOCA_RDMA_OPCODE_RECV_SEND_WITH_IMM
DOCA_RDMA_OPCODE_RECV_WRITE_WITH_IMM

enum doca_rdma_state

Possible states for doca_rdma

Values

DOCA_RDMA_STATE_RESET = 0
DOCA_RDMA_STATE_INIT
DOCA_RDMA_STATE_CONNECTED
DOCA_RDMA_STATE_ERROR

enum doca_rdma_transport_type

Available transport types for RDMA

Values

DOCA_RDMA_TRANSPORT_RC
RC transport
DOCA_RDMA_TRANSPORT_DC
DC transport, currently not supported

__DOCA_EXPERIMENTAL doca_ctx *doca_rdma_as_ctx (doca_rdma *rdma)

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

Parameters

rdma
RDMA instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

doca_error_t doca_rdma_connect (doca_rdma *rdma, const void *remote_rdma_conn_details, size_t remote_rdma_conn_details_size)

Connect to remote doca_rdma peer. Can only be called after calling doca_ctx_start().

Parameters

rdma
Pointer doca_rdma to export connection details for.

remote_rdma_conn_details
Exported doca_rdma_conn_details object from remote peer.
remote_rdma_conn_details_size
Size of remote doca_rdma_conn_details object.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if any of the parameters is null.
- DOCA_ERROR_BAD_STATE - if context was not started or rdma instance is already connected.
- DOCA_ERROR_CONNECTION_ABORTED - if connection failed or connection string was corrupted.

Description

Note:

stopping and restarting an RDMA context require calling doca_rdma_export() & doca_rdma_connect() again.

doca_error_t doca_rdma_create (doca_rdma **rdma)

Create a DOCA RDMA instance.

Parameters

rdma
Pointer to pointer to be set to point to the created doca_rdma instance.

Returns

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

- DOCA_ERROR_INVALID_VALUE - rdma argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to allocate resources.
- DOCA_ERROR_INITIALIZATION - failed to initialize rdma.

doca_error_t doca_rdma_destroy (doca_rdma *rdma)

Destroy a DOCA RDMA instance.

Parameters

rdma
Pointer to instance to be destroyed.

Returns

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

- DOCA_ERROR_INVALID_VALUE - rdma argument is a NULL pointer.
- DOCA_ERROR_BAD_STATE - the associated ctx was not stopped before calling doca_rdma_destroy().

doca_error_t doca_rdma_export (const doca_rdma *rdma, const void **local_rdma_conn_details, size_t *local_rdma_conn_details_size)

Export doca_rdma connection details object The doca_rdma_conn_details are used in doca_rdma_connect(). Can only be called after calling doca_ctx_start().

Parameters

rdma
Pointer doca_rdma to export connection details for.
local_rdma_conn_details
Exported doca_rdma_conn_details object.
local_rdma_conn_details_size
Size of exported doca_rdma_conn_details object.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if any of the parameters is null.
- DOCA_ERROR_BAD_STATE - if called before calling ctx_start().

Description


Note:

stopping and restarting an RDMA context require calling doca_rdma_export() & doca_rdma_connect() again.

doca_error_t doca_rdma_get_gid (doca_devinfo *devinfo, uint32_t start_index, uint32_t num_entries, doca_rdma_gid *gid_array)

Parameters

devinfo
The DOCA device information

start_index
The first gid index of interest

num_entries
The number of desired gid indicies

gid_array
A 'struct doca_rdma_gid' array of size 'num_entries', that on success will hold the desired gids. Note that it is the user's responsibility to provide an array with enough entries to prevent data corruption

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

Description

Get gids for a specific device by index and number of entries.

doca_error_t doca_rdma_get_gid_index (const doca_rdma *rdma, uint32_t *gid_index)

Get GID index from doca_rdma. Get the current GID index set for doca_rdma.

Parameters

rdma
doca_rdma context to get the property from.

gid_index
GID index used in doca_rdma.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_gid_table_size (doca_devinfo *devinfo, uint32_t *gid_table_size)

Parameters

devinfo
The DOCA device information

gid_table_size
The gid table size for the given devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

Description

Get the gid table size for a specific device.

doca_error_t doca_rdma_get_grh_enabled (const doca_rdma *rdma, bool *grh_enabled)

Get GRH setting from doca_rdma. Get the current GRH setting for doca_rdma.

Parameters

rdma
doca_rdma context to get the property from.

grh_enabled
GRH setting used in doca_rdma.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_max_buf_chain_len (const doca_devinfo *devinfo, doca_rdma_job_types job_type, doca_rdma_transport_type transport_type, uint32_t *max_buf_chain_len)

Parameters

devinfo
The DOCA device information

job_type
The relevant job type

transport_type
The relevant transport type

max_buf_chain_len
The maximal number of chained local buffers (containing data) that can be submitted in the given DOCA RDMA job for the given transport type and devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

Description

Get the maximal buffer chain length for a specific device, job type and transport type.

doca_error_t doca_rdma_get_max_message_size (const doca_devinfo *devinfo, uint32_t *max_message_size)

Get the maximal message size for a specific device.

Parameters

devinfo
The DOCA device information

max_message_size
The maximal message size for the given devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

doca_error_t doca_rdma_get_max_recv_queue_size (const doca_devinfo *devinfo, uint32_t *max_recv_queue_size)

Get the maximal recv queue size for a specific device.

Parameters

devinfo
The DOCA device information

max_recv_queue_size
The maximal recv queue size for the given devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

doca_error_t doca_rdma_get_max_send_queue_size (const doca_devinfo *devinfo, uint32_t *max_send_queue_size)

Parameters

devinfo
The DOCA device information

max_send_queue_size
The of the maximal send queue size for the given devinfo.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities.

Description

Get the maximal send queue size for a specific device.

doca_error_t doca_rdma_get_permissions (doca_rdma *rdma, uint32_t *permissions)

Get permissions property from doca_rdma. Returns the current permissions set for the doca_rdma_context. Can only be called after calling doca_ctx_dev_add().

Parameters

rdma
doca_rdma context to get the property from.

permissions
Bitwise combination of RDMA access flags set in context - see enum doca_access_flags

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_recv_buf_chain_len (const doca_rdma *rdma, uint32_t *recv_buf_chain_len)

Get the maximal receive buffer chain length from doca_rdma. Get the current receive buffer chain length set for doca_rdma. The returned value is the actual value being used and might differ from the size set by the user, as it may be increased.

Parameters

rdma
doca_rdma context to get the property from.

recv_buf_chain_len
recv_buf_chain_len used in doca_rdma.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_recv_queue_size (const doca_rdma *rdma, uint32_t *recv_queue_size)

Get recv queue size property from doca_rdma. Returns the current recv_queue_size set for the doca_rdma_context. The size returned is the actual size being used and might differ from the size set by the user, as the size may be increased.

Parameters

rdma
doca_rdma context to get the property from.

recv_queue_size
Recv queue size set in context.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_send_queue_size (const doca_rdma *rdma, uint32_t *send_queue_size)

Get send queue size property from doca_rdma. Returns the current send_queue_size set for the doca_rdma_context. The size returned is the actual size being used and might differ from the size set by the user, as the size may be increased.

Parameters

rdma
doca_rdma context to get the property from.

send_queue_size
Send queue size set in context.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_sl (const doca_rdma *rdma, uint32_t *sl)

Get SL (service level) from doca_rdma. Get the current SL set for doca_rdma.

Parameters

rdma
doca_rdma context to get the property from.

sl
SL used in doca_rdma.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_state (const doca_rdma *rdma, doca_rdma_state *state)

Get current state of doca_rdma. Returns the current state of the doca_rdma_context.

Parameters

rdma
doca_rdma context to get the state from.

state
State of doca_rdma context.

Returns

DOCA_SUCCESS - if state retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_transport_type (const doca_rdma *rdma, doca_rdma_transport_type *transport_type)

Get transport_type property from doca_rdma. Returns the current transport_type set for the doca_rdma_context.

Parameters

rdma
doca_rdma context to get the property from.

transport_type
Transport_type set in context.

Returns

DOCA_SUCCESS - if property retrieved successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if function was called before adding a device.

doca_error_t doca_rdma_get_transport_type_supported (const doca_devinfo *devinfo, doca_rdma_transport_type transport_type)

Check if DOCA RDMA supports given transport type for a specific device.

Parameters

devinfo
The DOCA device information

transport_type
Transport type to query support for.

Returns

DOCA_SUCCESS - in case the transport type is supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support the given transport type.

doca_error_t doca_rdma_job_get_supported (const doca_devinfo *devinfo, doca_rdma_job_types job_type)

Check if given device supports given doca_rdma job.

Parameters

devinfo
The DOCA device information

job_type
doca_rdma job type. See enum doca_rdma_job_types.

Returns

DOCA_SUCCESS - in case the job is supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_DRIVER - failed to query device capabilities
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support the given doca_rdma job.

doca_error_t doca_rdma_set_gid_index (doca_rdma *rdma, uint32_t gid_index)

Set GID index for doca_rdma. The value can be queried using doca_rdma_get_gid_index(). Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

gid_index
GID index to use in doca_rdma.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started or function was called before adding a device.

doca_error_t doca_rdma_set_grh_enabled (doca_rdma *rdma, bool grh_enabled)

Set whether to use GRH in connection. The value can be queried using doca_rdma_get_grh_enabled(). Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

grh_enabled
GRH setting to use in doca_rdma.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started or function was called before adding a device.
- DOCA_ERROR_NOT_SUPPORTED - if GRH setting is not supported for the device.

Description

If using IB device: If GRH is disabled, the address will rely on LID only. If GRH is enabled, the other side must also use GRH.

If using ETH device, GRH must be enabled.

doca_error_t doca_rdma_set_permissions (doca_rdma *rdma, uint32_t permissions)

Set rdma permissions for doca_rdma. The value can be queried using doca_rdma_get_permissions(). Can only be called after calling doca_ctx_dev_add() and before calling doca_ctx_start(). The supported permissions are the RDMA access flags.

Parameters

rdma
doca_rdma context to set the property for.

permissions
Bitwise combination of RDMA access flags - see enum doca_access_flags

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given or non-RDMA access flags were given.
- DOCA_ERROR_BAD_STATE - if context is already started or function was called before adding a device.

doca_error_t doca_rdma_set_recv_buf_chain_len (doca_rdma *rdma, uint32_t recv_buf_chain_len)

Set the maximal receive buffer chain length for doca_rdma. The length may be increased and the value in use can be queried using doca_rdma_get_recv_buf_chain_len(). Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

recv_buf_chain_len
recv_buf_chain_len to use in doca_rdma.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_rdma_set_recv_queue_size (doca_rdma *rdma, uint32_t recv_queue_size)

Set recv queue size property for doca_rdma. The value can be queried using doca_rdma_get_recv_queue_size(). Queue size will be rounded to the next power of 2. Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

recv_queue_size
Recv queue size to use in context.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_NOT_SUPPORTED - if the given size is not supported.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_rdma_set_send_queue_size (doca_rdma *rdma, uint32_t send_queue_size)

Set send queue size property for doca_rdma. The value can be queried using doca_rdma_get_send_queue_size(). Queue size will be rounded to the next power of 2. can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

send_queue_size
Send queue size to use in context.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_NOT_SUPPORTED - if the given size is not supported.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_rdma_set_sl (doca_rdma *rdma, uint32_t sl)

Set SL (service level) for doca_rdma. The value can be queried using doca_rdma_get_sl(). Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

sl
SL to use in doca_rdma.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_BAD_STATE - if context is already started.

doca_error_t doca_rdma_set_transport_type (doca_rdma *rdma, doca_rdma_transport_type transport_type)

Set transport type for doca_rdma. The value can be queried using doca_rdma_get_transport_type(). Can only be called before calling doca_ctx_start().

Parameters

rdma
doca_rdma context to set the property for.

transport_type
Transport type to use in context.

Returns

DOCA_SUCCESS - if property set successfully. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid parameter was given.
- DOCA_ERROR_NOT_SUPPORTED - if the given transport type is not supported.
- DOCA_ERROR_BAD_STATE - if context is already started.

#define DOCA_RDMA_GID_BYTE_LEN 16

GID length in bytes

2.24. RegEx engine

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

struct doca_regex_job_search

struct doca_regex_match

struct doca_regex_search_result

enum doca_regex_job_types

Available job types for RegEx.

Values

DOCA_REGEX_JOB_SEARCH = DOCA_ACTION_REGEX_FIRST+1
Default RegEx search mode

enum doca_regex_search_job_flags

Available job flags for RegEx.

Values

DOCA_REGEX_SEARCH_JOB_FLAG_HIGHEST_PRIORITY_MATCH = 1<<1
DOCA_REGEX_SEARCH_JOB_FLAG_STOP_ON_ANY_MATCH = 1<<2

enum doca_regex_status_flag

Response status flags

Values

DOCA_REGEX_STATUS_SEARCH_FAILED = 1

__DOCA_EXPERIMENTAL doca_ctx *doca_regex_as_ctx (doca_regex *regex)

Parameters

regex
The RegEx instance to convert. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert RegEx instance into context for use with workQ

doca_error_t doca_regex_create (doca_regex **regex)

Parameters

regex
Pointer to be populated with the address of the newly created RegEx context.

Returns

DOCA_SUCCESS - RegEx instance was created DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable to create required resources.

Description

Create a DOCA RegEx instance.

doca_error_t doca_regex_destroy (doca_regex *regex)

Parameters

regex
Instance to be destroyed, MUST NOT BE NULL.

Returns

DOCA_SUCCESS - RegEx instance was created DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Destroy DOCA RegEx instance.

Note:

The context must be stopped via a successful call to doca_ctx_stop before it can be safely destroyed.

doca_error_t doca_regex_get_failed_job_fallback_enabled (const doca_regex *regex, bool *enabled)

Parameters

regex
The RegEx engine.

enabled
Set to true when the feature is enabled, or false if it is disabled.

Returns

DOCA_SUCCESS - enabled is populated DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Is the failed job fallback feature enabled.

doca_error_t doca_regex_get_hardware_compiled_rules (const doca_regex *regex, void **rules_data, size_t *rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.

rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

Returns

DOCA_SUCCESS - rules_data and rules_data_size are populated.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Get the compiled rules data to be used by the hardware RegEx device.

doca_error_t doca_regex_get_hardware_supported (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Hardware acceleration is supported. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Hardware acceleration is NOT supported.

Description

Determine if a given device supports hardware accelerated RegEx searches.

doca_error_t doca_regex_get_hardware_uncompiled_rules (const doca_regex *regex, void **rules_data, size_t *rules_data_size)

Parameters

regex
The RegEx engine.
rules_data
Value to populate to hold a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.
rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

Returns

DOCA_SUCCESS - rules_data and rules_data_size are populated.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Get the un-compiled rules data to be used by the hardware RegEx device.

doca_error_t doca_regex_get_huge_job_emulation_overlap_size (const doca_regex *regex, uint16_t *nb_overlap_bytes)

Parameters

regex
The RegEx engine.
nb_overlap_bytes
Number of bytes of overlap in use.

Returns

DOCA_SUCCESS - nb_overlap_bytes is populated. DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

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

doca_error_t doca_regex_get_maximum_job_size (const doca_devinfo *devinfo, uint64_t *max_job_len)

Parameters

devinfo
Device to check.
max_job_len
Maximum supported job length.

Returns

DOCA_SUCCESS - max_job_len is populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device does not support RegEx.

Description

Determine the maximum job size supported by this device.

doca_error_t doca_regex_get_maximum_non_huge_job_size (const doca_devinfo *devinfo, uint64_t *max_job_len)

Parameters

devinfo
Device to check.

max_job_len
Maximum supported job length.

Returns

DOCA_SUCCESS - max_job_len is populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device does not support RegEx.

Description

Determine the maximum job size supported by this device without requiring the huge job emulation feature.

doca_error_t doca_regex_get_small_job_offload_threshold (const doca_regex *regex, uint16_t *threshold)

Parameters

regex
The RegEx engine.

threshold

Returns

DOCA_SUCCESS - threshold is populated. DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Get the "small jobs" threshold.

doca_error_t doca_regex_get_software_compiled_rules (const doca_regex *regex, void **rules_data, size_t *rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.

rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

Returns

DOCA_SUCCESS - rules_data and rules_data_size are populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Get the compiled rules data to be used by the software RegEx device.

doca_error_t doca_regex_get_software_supported (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Software support is available. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Software support is NOT available.

Description

Determine if a given device supports software based RegEx searches.

doca_error_t doca_regex_get_software_uncompiled_rules (const doca_regex *regex, void **rules_data, size_t *rules_data_size)

Parameters

regex
The RegEx engine.
rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.
rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

Returns

DOCA_SUCCESS - rules_data and rules_data_size are populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Get the un-compiled rules data to be used by the software RegEx device.

doca_error_t doca_regex_get_workq_matches_memory_pool_size (const doca_regex *regex, uint32_t *pool_size)

Parameters

regex
The RegEx engine.

pool_size
Number of items that will be available to each workq.

Returns

DOCA_SUCCESS - pool_size is populated. DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Get the size of work queue pool attached to workq for use with the RegEx engine.

doca_error_t doca_regex_is_supported (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Device can be used with doca_regex. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device cannot be used with doca_regex.

Description

Determine if a given device is suitable for use with doca_regex.

doca_error_t doca_regex_job_get_supported (const doca_devinfo *devinfo, doca_regex_job_types job_type)

Parameters

devinfo
Device to check.

job_type
Job type to check.

Returns

DOCA_SUCCESS - Job type is supported with the given device. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Job type is NOT supported with the given device.

Description

Determine if the given job type is supported for the given device.

doca_error_t doca_regex_search_job_flag_get_highest_priority_match_s (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Search jobs support highest priority match when using the given device. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - highest priority match is NOT supported when using the given device.

Description

Determine if 'highest priority' match is supported for the given device when submitting doca_regex_job_search jobs.

doca_error_t doca_regex_search_job_flag_get_stop_on_any_match_supp (const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Search jobs support stop on any match when using the given device. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - stop on any match is NOT supported when using the given device.

Description

Determine if 'stop on any' match is supported for the given device when submitting doca_regex_job_search jobs.

doca_error_t doca_regex_set_failed_job_fallback_enabled (doca_regex *regex, bool enabled)

Parameters

regex
The RegEx engine.
enabled
Specify true to enable the feature, false to disable it.

Returns

DOCA_SUCCESS - Property was successfully set DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description

Enable the ability to automatically migrate a job which was executed on the hardware device and subsequently failed to be re-executed on the software device. This is useful if a hardware limitation prevents a job from executing to completion.

Note:

This feature requires both a hardware and a software device to be available. Validated during context start.

doca_error_t doca_regex_set_hardware_compiled_rules (doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
An opaque blob of rules data which is provided to the hardware device.

rules_data_size
Size of the blob.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the compiled rules data to be used by the hardware RegEx device.

Note:

- This property is mutually exclusive with hardware un-compiled rules. This property mandates that a hardware device will be attached before the context is started. If no hardware device will be provided you should not specify hardware rules, or explicitly

clear them by setting them to NULL. This will be validated as part of starting the context

- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.

doca_error_t doca_regex_set_hardware_uncompiled_rules (doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
An opaque blob of rules data which will be compiled by the engine into compiled rules data which is then provided to the hardware device.

rules_data_size
Size of the blob.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the un-compiled rules data to be used by the hardware RegEx device.

Note:

- This property is mutually exclusive with hardware compiled rules. This property mandates that a hardware device will be attached before the context is started. If no hardware device will be provided you should not specify hardware rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context

- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.
- Rules compilation takes place during context start.

doca_error_t doca_regex_set_huge_job_emulation_overlap_size (doca_regex *regex, uint16_t nb_overlap_bytes)

Parameters

regex
The RegEx engine.

nb_overlap_bytes
Number of bytes of overlap to use.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description

Set the size of overlap to use when a job exceeds a devices maximum search size. Defaults to 0 (no overlap)

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

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

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

Note:

The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context

doca_error_t doca_regex_set_in_order_responses_enabled (doca_regex *regex, bool enabled)

Parameters

regex
The RegEx engine.

enabled
Set to true to ensure in order responses, false to turn off feature.

Returns

DOCA_SUCCESS - Property was successfully set DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description

Configure the driver so that each QP will return job responses in the same order that the jobs were sent.

Note:

Enabling in order reponses may negatively impact performance as the hardware device may process some jobs quicker than others but software will not have access to these responses until previous jobs in the queue are completed. This feature is disabled by default and is recommended to only be enabled if it is a hard requirement for a given use-case.

doca_error_t doca_regex_set_small_job_offload_threshold (doca_regex *regex, uint16_t threshold)

Parameters

regex
The RegEx engine.

threshold
Threshold job size in bytes.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description

Define a threshold for "small jobs". For scenarios where small jobs cause poor performance using the hardware RegEx device these can instead be redirected to the software device. Set this to a value > 0 to enable the feature. Set this value to 0 to disable the feature. Defaults to 0 (disabled)

Note:

This feature requires both a hardware and a software device to be available. The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context

doca_error_t doca_regex_set_software_compiled_rules (doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
An opaque blob of rules data which is provided to the software device.
rules_data_size
Size of the blob.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the compiled rules data to be used by the software RegEx device.

Note:

- This property is mutually exclusive with software un-compiled rules. This property mandates that a software device will be attached before the context is started. If no software device will be provided you should not specify software rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context
- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.

doca_error_t doca_regex_set_software_uncompiled_rules (doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters

regex
The RegEx engine.
rules_data
An opaque blob of rules data which will be compiled by the engine into compiled rules data which is then provided to the software device.
rules_data_size
Size of the blob

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the un-compiled rules data to be used by the software RegEx device.

Note:

- This property is mutually exclusive with software compiled rules. This property mandates that a software device will be attached before the context is started. If no software device will be provided you should not specify software rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context
- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.
- Rules compilation takes place during context start.

doca_error_t doca_regex_set_workq_matches_memory_pool_size (doca_regex *regex, uint32_t pool_size)

Parameters

regex
The RegEx engine.

pool_size
Number of items to have available to each workq.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure: DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description

Each work queue attached to the RegEx engine gets a pool allocator for matches. Set this value to set the maximum number of matches that can be stored for a given workq.

Note:

The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context

2.25. RegEx engine memory pool

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

__DOCA_EXPERIMENTAL doca_regex_mempool *doca_regex_mempool_create (size_t elem_size, size_t nb_elems)

Parameters

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

nb_elems
Number of element stored in the memory pool.

Returns

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

Description

Create a memory pool.

Note:

Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL void doca_regex_mempool_destroy (doca_regex_mempool *pool)

Parameters

pool

Memory pool to be destroyed. Must not be NULL.

Description

Destroy a memory pool and all objects it owned.

Note:

all pointers to elements in this pool must be cleared before this call. Failure to do socmay result in undefined behaviour.

__DOCA_EXPERIMENTAL size_t doca_regex_mempool_get_capacity (const doca_regex_mempool *pool)

Parameters

pool

The pool to query

Returns

The capacity / size of the pool

Description

Get the capacity of a mempool

__DOCA_EXPERIMENTAL size_t doca_regex_mempool_get_element_size (const doca_regex_mempool *pool)

Parameters

pool
The pool to query

Returns

The size available for storage of an element within the pool

Description

Get the size of element which can be stored in this pool

__DOCA_EXPERIMENTAL size_t doca_regex_mempool_get_free_count (const doca_regex_mempool *pool)

Parameters

pool
The pool to query

Returns

The number of free elements

Description

Get the number of elements available for use within the pool

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

Parameters

pool
Memory pool to fetch an object from.

n
Index of the object to be retrieved

Returns

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

Description

Directly access an object in the mempool by index.

Note:

- this function does not care if the object is in use or free.
- Supports single producer and single consumer only.

__DOCA_EXPERIMENTAL void *doca_regex_mempool_get_obj (doca_regex_mempool *pool)

Parameters

pool
Pool from which to get a free object.

Returns

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

Description

Get an object from the memory pool.

Note:

Supports single producer and single consumer only.

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

Parameters

pool
Memory pool owning the object.

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

Returns

0 based index of element or a negative error code.

Description

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

Note:

Supports single producer and single consumer only.

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

Parameters

pool
Pool which created obj.

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

Description

Put an object back into the memory pool.

Note:

Supports single producer and single consumer only.

2.26. DOCA RMAX engine

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

struct doca_rmax_cpu_affinity_mask

Data structure to describe CPU mask for doca_rmax internal thread.

struct doca_rmax_in_stream_completion

Completion returned by input stream describing the incoming packets.

struct doca_rmax_job_rx_data

Receive data job.

struct doca_rmax_stream_error

Detailed completion error information.

enum doca_rmax_action_type

Action types for doca jobs.

Values

DOCA_RMAX_ACTION_TYPE_RX_DATA = DOCA_ACTION_RMAX_FIRST+1
Input packets data. This action does not originate from a submitted job.

enum doca_rmax_in_stream_scatter_type

Incoming packet scatter mode, used by input stream.

Values

DOCA_RMAX_IN_STREAM_SCATTER_TYPE_RAW = 0
Store raw packet data including network headers
DOCA_RMAX_IN_STREAM_SCATTER_TYPE_ULP
Store User-Level Protocol only data (discard network header up to L4)
DOCA_RMAX_IN_STREAM_SCATTER_TYPE_PAYLOAD
Store payload data only (all headers will be discarded)

enum doca_rmax_in_stream_ts_fmt_type

Input packet timestamp format (timestamp, when packet was received).

Values

DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_COUNTER = 0
Raw number written by HW, representing the HW clock
DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_NANO
Time in nanoseconds
DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_SYNCED
Time in nanoseconds, synced with PTP grandmaster

enum doca_rmax_in_stream_type

Type of input stream.

Values

DOCA_RMAX_IN_STREAM_TYPE_GENERIC = 0
Generic stream
DOCA_RMAX_IN_STREAM_TYPE_RTP_2110
SMPTE ST 2110 stream

typedef uint64_t doca_rmax_cpu_mask_t

CPU bitmask container

doca_error_t doca_rmax_flow_attach (const doca_rmax_flow *flow, const doca_rmax_in_stream *stream)

Attach a flow to a stream.

Parameters

flow
Flow to operate on
stream
The context for attaching a flow

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_SHUTDOWN - library shutdown in a process.
- DOCA_ERROR_UNEXPECTED - unexpected issue.

doca_error_t doca_rmax_flow_create (doca_rmax_flow **flow)

Create a steering flow for input stream to filter incoming data flow by match criteria.

Parameters

flow
The flow created for input stream. Non NULL upon success, NULL otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - unable to allocate memory.

doca_error_t doca_rmax_flow_destroy (doca_rmax_flow *flow)

Destroy a steering flow.

Parameters

flow
Flow to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_detach (const doca_rmax_flow *flow, const doca_rmax_in_stream *stream)

Detach a flow from a stream.

Parameters

flow
Flow to operate on

stream
The context for detaching a flow

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_SHUTDOWN - library shutdown in a process.
- DOCA_ERROR_UNEXPECTED - unexpected issue.

doca_error_t doca_rmax_flow_set_dst_ip (doca_rmax_flow *flow, const in_addr *ip)

Set the destination IP filter for the flow.

Parameters

flow
Flow to operate on

ip
Destination IPv4 address

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_dst_port (doca_rmax_flow *flow, uint16_t port)

Set the destination port filter for the flow.

Parameters

flow
Flow to operate on

port
Destination port number, non-zero

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_src_ip (doca_rmax_flow *flow, const in_addr *ip)

Set the source IP filter for the flow.

Parameters

flow
Flow to operate on

ip
Source IPv4 address

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_src_port (doca_rmax_flow *flow, uint16_t port)

Set the source port filter for the flow.

Parameters

flow
Flow to operate on

port
Source port number. If zero then any source port is accepted.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_tag (doca_rmax_flow *flow, uint32_t tag)

Set the tag for the flow.

Parameters

flow
Flow to operate on

tag
Non-zero tag

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_get_cpu_affinity_mask (doca_rmax_cpu_affinity_mask *mask)

Get affinity mask for the internal Rivermax thread.

Parameters

mask

Affinity mask. CPU is included in affinity mask if the corresponding bit is set. If CPU affinity mask is unset return value is zeroed.

Returns

DOCA_SUCCESS - always.

doca_error_t doca_rmax_get_ptp_clock_supported (const doca_devinfo *devinfo)

Query PTP clock capability for device.

Parameters

devinfo

The device to query

Returns

DOCA_SUCCESS - PTP clock is supported. DOCA_ERROR_NOT_SUPPORTED - PTP clock is not supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv4 address from the OS

__DOCA_EXPERIMENTAL doca_ctx *doca_rmax_in_stream_as_ctx (doca_rmax_in_stream *stream)

Convert a DOCA RMAX input stream to DOCA context.

Parameters

stream
The context to be converted

Returns

The matching doca_ctx instance in case of success, NULL otherwise.

Description

DOCA RMAX stream supports all stream operations: create/start/stop/destroy. Only one device and one workq must be attached to a stream.

doca_error_t doca_rmax_in_stream_create (doca_rmax_in_stream **stream)

Create a DOCA RMAX input stream context.

Parameters

stream
The input stream context created for the DOCA RMAX. Non NULL upon success, NULL otherwise.

Returns

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

- DOCA_ERROR_INVALID_VALUE - stream argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_rmax_in_stream.
- DOCA_ERROR_INITIALIZATION - failed to initialise DOCA context.

Description

Create input stream.

doca_error_t doca_rmax_in_stream_destroy (doca_rmax_in_stream *stream)

Destroy a DOCA input stream context.

Parameters

stream
The context to be destroyed

Returns

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

- DOCA_ERROR_INVALID_VALUE - stream argument is a NULL pointer.

Description

Free all allocated resources associated with a DOCA RMAX input stream.

doca_error_t doca_rmax_in_stream_get_elements_count (const doca_rmax_in_stream *stream, uint32_t *value)

Get number of elements in the stream buffer.

Parameters

stream
The input stream to query.
value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This value can differ from value set by doca_rmax_in_stream_set_elements_count if the argument is not a power of two.

doca_error_t doca_rmax_in_stream_get_max_packets (const doca_rmax_in_stream *stream, uint32_t *value)

Get maximal number of packets that input stream must return in read event.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_in_stream_get_memblk_size (const doca_rmax_in_stream *stream, size_t *value)

Get size of memory block(s).

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Size of memory block (array of sizes for multiple memblks, the number of memory blocks in stream is more than one).

doca_error_t doca_rmax_in_stream_get_memblk_stride_size (const doca_rmax_in_stream *stream, uint16_t *value)

Get stride size(s).

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Stride size of memory block (array of stride sizes for multiple memory blocks).

doca_error_t doca_rmax_in_stream_get_memblks_count (const doca_rmax_in_stream *stream, uint32_t *value)

Get number of configured memory blocks.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Amount of memblks is equal to the number of segments into which the incoming packet will be divided.

doca_error_t doca_rmax_in_stream_get_min_packets (const doca_rmax_in_stream *stream, uint32_t *value)

Get minimal number of packets that input stream must return in read event.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_in_stream_get_scatter_type (const doca_rmax_in_stream *stream, doca_rmax_in_stream_scatter_type * *value)

Get the type of packet's data scatter.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

See enum doca_rmax_in_stream_scatter_type.

doca_error_t doca_rmax_in_stream_get_timeout_us (const doca_rmax_in_stream *stream, int *value)

Get receive timeout.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

Description

The number of usecs that library would do busy wait (polling) for reception of at least `min_packets` number of packets.

doca_error_t doca_rmax_in_stream_get_timestamp_format (const doca_rmax_in_stream *stream, doca_rmax_in_stream_ts_fmt_type * *value)

Get stream timestamp format.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

See enum doca_rmax_in_stream_ts_fmt_type

doca_error_t doca_rmax_in_stream_get_type (const doca_rmax_in_stream *stream, doca_rmax_in_stream_type * *value)

Get input stream type.

Parameters

stream
The input stream to query.
value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_in_stream_memblk_desc_get_max_size (const doca_rmax_in_stream *stream, uint16_t *value)

Get maximal packet segment sizes.

Parameters

stream
The input stream to query.
value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Array of maximal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer.

doca_error_t doca_rmax_in_stream_memblk_desc_get_min_size (const doca_rmax_in_stream *stream, uint16_t *value)

Get minimal packet segment sizes.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Array of minimal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer.

doca_error_t doca_rmax_in_stream_memblk_desc_set_max_size (doca_rmax_in_stream *stream, const uint16_t *value)

Set maximal packet segment sizes.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Array of maximal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer. Must be set before starting the stream context.

doca_error_t doca_rmax_in_stream_memblk_desc_set_min_size (doca_rmax_in_stream *stream, const uint16_t *value)

Set minimal packet segment sizes.

Parameters

stream
The input stream to write property.
value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Array of minimal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer. Default: 0.

doca_error_t doca_rmax_in_stream_set_elements_count (doca_rmax_in_stream *stream, uint32_t value)

Set number of elements in the stream buffer.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Must be set before starting the stream context. See also doca_rmax_in_stream_get_elements_count.

doca_error_t doca_rmax_in_stream_set_max_packets (doca_rmax_in_stream *stream, uint32_t value)

Set maximal number of packets that input stream must return in read event.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Default: 1024.

doca_error_t doca_rmax_in_stream_set_memblk (doca_rmax_in_stream *stream, doca_buf *buf)

Set memory buffer(s).

Parameters

stream
The input stream to write property.

buf
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Memory buffer (or head of linked list of memory buffers) for storing received data. The length of linked list must be the same as number of memory blocks configured. Must be set before starting the stream context.

doca_error_t doca_rmax_in_stream_set_memblks_count (doca_rmax_in_stream *stream, uint32_t value)

Set number of configured memory blocks.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Amount of memblks is equal to the number of segments into which the incoming packet will be divided. Default: 1. Valid values: 1 and 2.

doca_error_t doca_rmax_in_stream_set_min_packets (doca_rmax_in_stream *stream, uint32_t value)

Set minimal number of packets that input stream must return in read event.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Default: 0.

doca_error_t doca_rmax_in_stream_set_scatter_type (doca_rmax_in_stream *stream, doca_rmax_in_stream_scatter_type value)

Set the type of packet's data scatter.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

See enum doca_rmax_in_stream_scatter_type. Default: DOCA_RMAX_IN_STREAM_SCATTER_TYPE_RAW.

doca_error_t doca_rmax_in_stream_set_timeout_us (doca_rmax_in_stream *stream, int value)

Set receive timeout.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The number of usecs that library would do busy wait (polling) for reception of at least `min_packets` number of packets.

Default: 0.

doca_error_t doca_rmax_in_stream_set_timestamp_format (doca_rmax_in_stream *stream, doca_rmax_in_stream_ts_fmt_type value)

Set stream timestamp format.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

See enum doca_rmax_in_stream_ts_fmt_type Default: DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_COUNTER.

doca_error_t doca_rmax_in_stream_set_type (doca_rmax_in_stream *stream, doca_rmax_in_stream_type value)

Set input stream type.

Parameters

stream
The input stream to write property.

value
Property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Default: DOCA_RMAX_IN_STREAM_TYPE_GENERIC.

doca_error_t doca_rmax_init (void)

DOCA RMAX library initalization.

Returns

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

- DOCA_ERROR_UNSUPPORTED_VERSION - unsupported Rivermax library version.
- DOCA_ERROR_INITIALIZATION - Rivermax initialization failed.
- DOCA_ERROR_NO_MEMORY - unable to allocate memory.
- DOCA_ERROR_NOT_FOUND - there are no supported devices.
- DOCA_ERROR_NOT_SUPPORTED - invalid or missing Rivermax license.
- DOCA_ERROR_UNEXPECTED - unexpected issue.

Description

This function initializes the DOCA RMAX global resources. This function must be called after doca_rmax_set_cpu_affinity_mask and before any other DOCA RMAX library call.

__DOCA_EXPERIMENTAL void doca_rmax_interrupt (void)

Interrupt the currently executing DOCA RMAX function, if any.

doca_error_t doca_rmax_release (void)

Uninitialize DOCA RMAX library.

Returns

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

- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_IN_USE - library is in use.

Description

This function cleans up the DOCA RMAX resources. No DOCA RMAX function may be called after calling this function.

doca_error_t doca_rmax_set_clock (doca_dev *dev)

Set the device to use for obtaining PTP time.

Parameters

dev

Device to use for obtaining the PTP time.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_FOUND - there is no IPv4 address associated with this device.
- DOCA_ERROR_NOT_SUPPORTED - PTP clock is not supported by device.
- DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv4 address from the OS
- DOCA_ERROR_UNEXPECTED - unexpected issue.

Description

The device must have PTP clock capability, see doca_rmax_get_ptp_clock_supported.

doca_error_t doca_rmax_set_cpu_affinity_mask (const doca_rmax_cpu_affinity_mask *mask)

Set affinity mask for the internal Rivermax thread.

Parameters

mask

Affinity mask. CPU is included in affinity mask if the corresponding bit is set. By default affinity mask is not set, so internal thread can run on any CPU core.

Returns

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

- DOCA_ERROR_INVALID_VALUE - invalid affinity mask provided.

Description

Must be called before doca_rmax_init().

#define DOCA_RMAX_CPU_SETSIZE 1024

maximum CPU set size

#define DOCA_RMAX_NCPUBITS (8 * sizeof(doca_rmax_cpu_mask_t))

number of CPU bits per one cpu mask element

2.27. engine

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

struct doca_sha_job

struct doca_sha_partial_job

enum doca_sha_job_flags

Flag enum for SHA job.

Values

DOCA_SHA_JOB_FLAGS_NONE = 0
Default flag for all SHA job.
DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL
Only useful for doca_sha_partial_job.

enum doca_sha_job_type

Doca sha action type enums, used to specify sha job types.

Values

DOCA_SHA_JOB_SHA1 = DOCA_ACTION_SHA_FIRST+1
DOCA_SHA_JOB_SHA256
DOCA_SHA_JOB_SHA512
DOCA_SHA_JOB_SHA1_PARTIAL
DOCA_SHA_JOB_SHA256_PARTIAL
DOCA_SHA_JOB_SHA512_PARTIAL

__DOCA_EXPERIMENTAL doca_ctx *doca_sha_as_ctx (doca_sha *ctx)

Parameters

ctx
SHA instance. This must remain valid until after the context is no longer required.

Returns

Non NULL - doca_ctx object on success. Error:

- NULL.

Description

Convert sha instance into context for use with workQ

doca_error_t doca_sha_create (doca_sha **ctx)

Parameters

ctx
Instance pointer to be created, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - not enough memory for allocation.
- DOCA_ERROR_NOT_SUPPORTED - the required engine is not supported.

Description

Create a DOCA SHA instance.

doca_error_t doca_sha_destroy (doca_sha *ctx)

Parameters

ctx
Instance to be destroyed, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call doca_ctx_stop().

Description

Destroy DOCA SHA instance.

doca_error_t doca_sha_get_hardware_supported (const doca_devinfo *devinfo)

Parameters

devinfo
The DOCA device information

Returns

DOCA_SUCCESS - in case of success, it is a hardware_based engine. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description

Get DOCA SHA operating mode: hardware or openssl_fallback. First to use doca_sha_job_get_supported() to decide whether the devinfo support doca_sha engine. Second to use doca_sha_get_hardware_supported() to decide whether this doca_sha engine is hardware_based or openssl_fallback_based.

doca_error_t doca_sha_get_max_list_buf_num_elem (const doca_devinfo *devinfo, uint32_t *max_list_num_elem)

Parameters

devinfo
The DOCA device information.

max_list_num_elem
The maximum supported number of elements in a given DOCA linked-list buffer, such that 1 indicates no linked-list buffer support.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description

Get the maximum supported number of elements in a given DOCA linked-list buffer for SHA job.

doca_error_t doca_sha_get_max_src_buffer_size (const doca_devinfo *devinfo, uint64_t *max_buffer_size)

Parameters

devinfo
The DOCA device information

max_buffer_size
The max buffer size for DOCA SHA operation in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description

Get maximum source buffer size for DOCA SHA job.

doca_error_t doca_sha_get_min_dst_buffer_size (const doca_devinfo *devinfo, doca_sha_job_type job_type, uint32_t *min_buffer_size)

Parameters

devinfo
The DOCA device information

job_type
doca_sha job type. See enum doca_sha_job_type.

min_buffer_size
The min buffer size for DOCA SHA operation in bytes.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description

Get minimum result buffer size for DOCA SHA job.

doca_error_t doca_sha_job_get_supported (const doca_devinfo *devinfo, doca_sha_job_type job_type)

Parameters

devinfo
The DOCA device information

job_type
SHA job_type available through this device. see enum doca_sha_job_type.

Returns

DOCA_SUCCESS - in case device supports job_type. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this SHA job.

Description

Check if given device is capable of excuting a specific SHA job.

doca_error_t doca_sha_partial_session_copy (doca_sha *ctx, doca_workq *workq, const doca_sha_partial_session *from_session, doca_sha_partial_session *to_session)

Parameters

ctx
SHA instance.
workq
Workq instance.
from_session
The source sha_partial_session object to be copied.
to_session
The dest sha_partial_session object.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - received corrupted input.

Description

Copy the sha_partial session state from "from_session" to "to_session". This is useful if large amounts of data to be sha-calculated which share the same header segments, and only differ in the tail last few bytes. For example, we have two msgs: msg_0 = {header, tail_0}, msg_1 = {header, tail_1}, both of them need to be processed in sha_partial job.

when without session_copy functionality: we will do sha_calculation for msg_0 as: doca_sha_partial_session_create(ctx, workq, &session_0); doca_workq_submit(workq, &job_0_for_header); doca_workq_progress_retrieve(workq, &event,

DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_workq_submit(workq, &job_0_for_tail_0); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_sha_partial_session_destroy(ctx, workq, session_0); then, we will do sha_calculation for msg_1 as: doca_sha_partial_session_create(ctx, workq, &session_1); doca_workq_submit(workq, &job_1_for_header); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_workq_submit(workq, &job_1_for_tail_1); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_sha_partial_session_destroy(ctx, workq, session_1);

Obviously, the same data "header" is calculated twice!

when utilising the session_copy functionality, we will do sha_calculation for msg_0 as: doca_sha_partial_session_create(ctx, workq, &session_0); doca_workq_submit(workq, &job_0_for_header); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); do a session_copy: doca_sha_partial_session_create(ctx, workq, &session_1); doca_sha_partial_session_copy(ctx, workq, session_0, session_1); continue to finish msg_0: doca_workq_submit(workq, &job_0_for_tail_0); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_sha_partial_session_destroy(ctx, workq, session_0); continue to finish msg_1: doca_workq_submit(workq, &job_1_for_tail_1); doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); doca_sha_partial_session_destroy(ctx, workq, session_1);

doca_error_t doca_sha_partial_session_create (doca_sha *ctx, doca_workq *workq, doca_sha_partial_session **session)

Parameters

ctx
SHA instance.

workq
Workq instance.

session
Instance pointer to be created, MUST NOT BE NULL.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

- DOCA_ERROR_NO_MEMORY - the resource is exhausted for now, please wait the existing jobs to finish or call doca_sha_partial_session_destroy() to release the allocated session objects.

Description

Get a session object used for sha_partial calculation. For the same sha_partial job, user need to keep using the same sha_partial_session. A session object can only be used for the same sha_ctx and workq. It cannot be shared between different sha_ctx or different workq.

doca_error_t doca_sha_partial_session_destroy (doca_sha *ctx, doca_workq *workq, doca_sha_partial_session *session)

Parameters

ctx
SHA instance.
workq
Workq instance.
session
A sha_partial_session object to be released.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

Release the session object used for sha_partial calculation. Please make sure the session to be released belonging to the specific sha_ctx and workq.

#define DOCA_SHA1_BYTE_COUNT 20

User response buffer length should be >= the following specified length, depending on the SHA function. Both SHA1_PARTIAL and SHA1 need 20bytes. Both SHA256_PARTIAL and SHA256 need 32bytes. Both SHA512_PARTIAL and SHA512 need 64bytes. SHA1 calculation hex-format result length.

#define DOCA_SHA256_BYTE_COUNT 32

SHA256 calculation hex-format result length.

#define DOCA_SHA512_BYTE_COUNT 64

SHA512 calculation hex-format result length.

2.28. Telemetry Service Library

DOCA lib for exporting events to the telemetry service.

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

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

See also:

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

Limitations:

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

enum doca_telemetry_ipc_status_t

DOCA telemetry IPC status.

Values

DOCA_TELEMETRY_IPC_STATUS_FAILED = -1
DOCA_TELEMETRY_IPC_STATUS_CONNECTED
DOCA_TELEMETRY_IPC_STATUS_DISABLED

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_error_t doca_telemetry_check_ipc_status (doca_telemetry_source *doca_source, doca_telemetry_ipc_status_t *status)

Return status of IPC transport.

Parameters

doca_source

Input doca source.

status

DOCA_TELEMETRY_IPC_STATUS_FAILED - if IPC is not connected.
DOCA_TELEMETRY_IPC_STATUS_CONNECTED - if IPC is connected.
DOCA_TELEMETRY_IPC_STATUS_DISABLED - if IPC is disabled from config.
if return is DOCA_SUCCESS then status can be one of the following

Returns

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

- DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.
- DOCA_TELEMETRY_IPC_STATUS_FAILED - if IPC is not connected.
- DOCA_TELEMETRY_IPC_STATUS_CONNECTED - if IPC is connected.
- DOCA_TELEMETRY_IPC_STATUS_DISABLED - if IPC is disabled from config.

doca_error_t doca_telemetry_field_create (doca_telemetry_field **field)

Create new telemetry field.

Parameters

field

Pointer to the newly allocated field.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry field.

doca_error_t doca_telemetry_field_destroy (doca_telemetry_field *field)

Destroy field previously created by doca_telemetry_field_create().

Parameters

field
Pointer to the field.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

__DOCA_EXPERIMENTAL void doca_telemetry_field_set_array_length (doca_telemetry_field *field_info, uint16_t len)

Set doca telemetry field length.

Parameters

field_info
Pointer to doca telemetry field.
len
Field length.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter(s) or invalid length (zero).

Description

Note:

If using single-value type (i.e char) this should be 1.

__DOCA_EXPERIMENTAL void doca_telemetry_field_set_description (doca_telemetry_field *field_info, const char *desc)

Set doca telemetry field description.

Parameters

field_info
Pointer to doca telemetry field.

desc
Field description.

Description

Note:

Passing a field_info value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_field_set_name (doca_telemetry_field *field_info, const char *name)

Set doca telemetry field name.

Parameters

field_info
Pointer to doca telemetry field.

name
Field name.

Description

Note:

Passing a field_info value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_field_set_type_name (doca_telemetry_field *field_info, const char *type)

Set doca telemetry field type.

Parameters

field_info
Pointer to doca telemetry field.

type
Field type.

Description

Note:

Please see DOCA_TELEMETRY_FIELD_TYPE_* for possible field types

Note:

Passing a field_info value of NULL will result in an undefined behavior.

doca_error_t doca_telemetry_get_timestamp (doca_telemetry_timestamp_t *timestamp)

Get timestamp in the proper format.

Parameters

timestamp
Timestamp value

Returns

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

- DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.

doca_error_t doca_telemetry_netflow_destroy (void)

Free the exporter memory and close the connection.

Returns

DOCA_SUCCESS - in case of success.

doca_error_t doca_telemetry_netflow_field_create (doca_telemetry_netflow_flowset_field **field)

Create new telemetry netflow field.

Parameters

field

Pointer to the newly allocated telemetry field.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow field.

doca_error_t doca_telemetry_netflow_field_destroy (doca_telemetry_netflow_flowset_field *field)

Destructor for DOCA netflow field.

Parameters

field

field to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if netflow_template is NULL.

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_field_set_length (doca_telemetry_netflow_flowset_field *field, uint16_t length)

Set doca telemetry netflow field length.

Parameters

field
Pointer to doca telemetry netflow field.

length
Field type.

Description

Note:

Passing a field value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_field_set_type (doca_telemetry_netflow_flowset_field *field, uint16_t type)

Set doca telemetry netflow field type.

Parameters

field
Pointer to doca telemetry netflow field.

type
Field type.

Description

Note:

Passing a field value of NULL will result in an undefined behavior.

doca_error_t doca_telemetry_netflow_flush (void)

Immediately flush the data of the DOCA internal Netflow source.

Returns

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

- DOCA_ERROR_BAD_STATE - if the netflow has not been started.

doca_error_t doca_telemetry_netflow_get_buffer_data_root (const char **path)

Get data root path.

Parameters

path
The buffer data root

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

doca_error_t doca_telemetry_netflow_get_buffer_size (uint64_t *size)

Get buffer size.

Parameters

size
The buffer size

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_netflow_get_file_write_max_age (doca_telemetry_timestamp_t *max_age)

Get file maximum age.

Parameters

max_age

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_netflow_get_file_write_max_size (size_t *size)

Get file maximum size.

Parameters

size

Maximum size of binary data file.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_netflow_get_ipc_sockets_dir (const char **path)

Get IPC socket directory.

Parameters

path

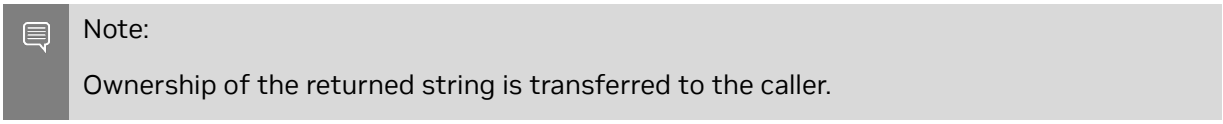
Path to a folder containing DOCA Telemetry Service (DTS) sockets.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description



doca_error_t doca_telemetry_netflow_init (uint16_t source_id)

Init exporter memory, set configs and open connection.

Parameters

source_id
Unique source ID.

Returns

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

- DOCA_ERROR_BAD_STATE - if the netflow has been initialized before this call.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.
- DOCA_ERROR_INITIALIZATION - failed to initialise netflow.

Description

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

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

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

Sending netflow records. Need to init first.

Parameters

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

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

nof_records
Records array size.

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

Returns

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

- DOCA_ERROR_BAD_STATE - if the netflow has not been initialized or the netflow has started.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:

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

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_buffer_data_root (const char *path)

Set buffer data root Default path is "/opt/mellanox/doca/services/telemetry/data/".

Parameters

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

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_buffer_size (uint64_t size)

Set buffer size Default value is 60000 bytes.

Parameters

size
Buffer size

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_collector_addr (const char *collector_addr)

Set collector address.

Parameters

collector_addr
User defined netflow collector's IP address.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_collector_port (uint16_t collector_port)

Set collector port. See DOCA_NETFLOW_DEFAULT_PORT for default value.

Parameters

collector_port
User defined netflow collector's port.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_file_write_enabled (void)

Enable file write file write is disabled by default.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_file_write_max_age (doca_telemetry_timestamp_t max_age)

Set file maximum age Default value is 1 hour.

Parameters

max_age

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

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_file_write_max_size (size_t size)

Set file maximum size Default value is 1MB.

Parameters

size

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

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_ipc_enabled (void)

Enable IPC IPC is disabled by default.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_ipc_sockets_dir (const char *path)

Set IPC socket directory. Default path is "/opt/mellanox/doca/services/telemetry/ipc_sockets".

Parameters

path

Path to a folder containing DOCA Telemetry Service (DTS) sockets.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_max_packet_size (uint16_t max_packet_size)

Set max packet size.

Parameters

max_packet_size
User defined netflow packet's max size.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_source_set_id (const char *source_id)

Set source id.

Parameters

source_id
Hostname or guid.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void doca_telemetry_netflow_source_set_tag (const char *source_tag)

Set source tag.

Parameters

source_tag
User defined data-file name prefix.

Description

Note:

This function should be called after doca_telemetry_netflow_init().

doca_error_t doca_telemetry_netflow_start (void)

Finalizes netflow setup.

Returns

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

- DOCA_ERROR_BAD_STATE - if the netflow has not been initialized or the netflow has started.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

doca_error_t doca_telemetry_netflow_template_add_field (doca_telemetry_netflow_template *netflow_template, doca_telemetry_netflow_flowset_field *field)

Add DOCA telemetry netflow field to netflow_template. The user loses the ownership of the field after a successful invocation of the function.

Parameters

netflow_template
Pointer to netflow_template.

field
DOCA Telemetry netflow field to add.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow field.

Description

Note:

field should NOT be passed to another group after calling this function.

doca_error_t doca_telemetry_netflow_template_create (doca_telemetry_netflow_template **netflow_template)

Create new telemetry netflow template.

Parameters

netflow_template
Pointer to the newly allocated telemetry netflow template.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow template.

doca_error_t doca_telemetry_netflow_template_destroy (doca_telemetry_netflow_template *netflow_template)

Destructor for DOCA netflow template.

Parameters

netflow_template
netflow template to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if netflow_template is NULL.

doca_error_t doca_telemetry_schema_add_type (doca_telemetry_schema *doca_schema, const char *new_type_name, doca_telemetry_type *type, doca_telemetry_type_index_t *type_index)

Add user-defined fields to create new type in DOCA schema. The users loses the ownership of the type after a successful invocation of the function.

Parameters

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

Returns

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

- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.
- DOCA_ERROR_INVALID_VALUE - If type name exists or any of the fields have invalid field type

doca_error_t doca_telemetry_schema_destroy (doca_telemetry_schema *doca_schema)

Destructor for DOCA schema.

Parameters

doca_schema
Schema to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if doca_schema is NULL.

doca_error_t doca_telemetry_schema_get_buffer_data_root (doca_telemetry_schema *doca_schema, const char **path)

Get data root path.

Parameters

doca_schema
Pointer to DOCA schema.
path
Path to a folder where the data and schema will be stored.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:

Ownership of the returned string is transferred to the caller.

doca_error_t doca_telemetry_schema_get_buffer_size (doca_telemetry_schema *doca_schema, uint64_t *size)

Get buffer size.

Parameters

doca_schema
Pointer to DOCA schema.

size
The buffer size

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_file_write_max_age (doca_telemetry_schema *doca_schema, doca_telemetry_timestamp_t *max_age)

Get file maximum age.

Parameters

doca_schema
Pointer to DOCA schema.

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_file_write_max_size (doca_telemetry_schema *doca_schema, size_t *size)

Get file maximum size.

Parameters

doca_schema
Pointer to DOCA schema.

size
Maximum size of binary data file.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_ipc_reconnect_time (doca_telemetry_schema *doca_schema, uint32_t *max_time)

Get IPC reconnect time in milliseconds.

Parameters

doca_schema
Pointer to DOCA schema.

max_time
Maximum reconnect time in milliseconds

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_ipc_reconnect_tries (doca_telemetry_schema *doca_schema, uint8_t *tries)

Get maximum IPC reconnect tries.

Parameters

doca_schema
Pointer to DOCA schema.

tries
Maximum reconnect tries

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_ipc_socket_timeout (doca_telemetry_schema *doca_schema, uint32_t *timeout)

Get IPC socket timeout in milliseconds.

Parameters

doca_schema
Pointer to ipc timeout attribute.

timeout
Maximum socket timeout in milliseconds

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t doca_telemetry_schema_get_ipc_sockets_dir (doca_telemetry_schema *doca_schema, const char **sockets_dir)

Get IPC socket directory.

Parameters

doca_schema
Pointer to DOCA schema.

sockets_dir
Path to a folder containing DOCA Telemetry Service (DTS) sockets.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:

Ownership of the returned string is transferred to the caller.

doca_error_t doca_telemetry_schema_init (const char *schema_name, doca_telemetry_schema **doca_schema)

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.

doca_schema
Pointer to DOCA schema, NULL on error.

Returns

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

- DOCA_ERROR_NO_MEMORY - failed to allocate doca_schema.
- DOCA_ERROR_INITIALIZATION - failed to initialise doca_schema.
- DOCA_ERROR_INVALID_VALUE - invalid input/output parameters.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_buffer_data_root (doca_telemetry_schema *doca_schema, const char *path)

Set buffer data root Default path is "/opt/mellanox/doca/services/telemetry/data/".

Parameters

doca_schema
Pointer to DOCA schema.

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

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_buffer_size (doca_telemetry_schema *doca_schema, uint64_t size)

Set buffer size Default value is 60000 bytes.

Parameters

doca_schema
Pointer to DOCA schema.

size
Buffer size

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_file_write_enabled (doca_telemetry_schema *doca_schema)

Enable file write file write is disabled by default.

Parameters

doca_schema

Pointer to DOCA schema.

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_file_write_max_age (doca_telemetry_schema *doca_schema, doca_telemetry_timestamp_t max_age)

Set file maximum age Default value is 1 hour.

Parameters

doca_schema

Pointer to DOCA schema.

max_age

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

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_file_write_max_size (doca_telemetry_schema *doca_schema, size_t size)

Set file maximum size Default value is 1MB.

Parameters

doca_schema

Pointer to DOCA schema.

size

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

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_ipc_enabled (doca_telemetry_schema *doca_schema)

Enable IPC IPC is disabled by default.

Parameters

doca_schema

Pointer to DOCA schema.

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_ipc_reconnect_time (doca_telemetry_schema *doca_schema, uint32_t max_time)

Set IPC reconnect time in milliseconds Time limit for reconnect attempts. If the limit is reached, the client is considered disconnected. Default value is 100 milliseconds.

Parameters

doca_schema
Pointer to DOCA schema.

max_time
Maximum reconnect time in milliseconds

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_ipc_reconnect_tries (doca_telemetry_schema *doca_schema, uint8_t tries)

Set maximum IPC reconnect tries. Number of reconnect attempts during reconnection period. Default value is 3 tries.

Parameters

doca_schema
Pointer to DOCA schema.

tries
Maximum reconnect tries

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_ipc_socket_timeout (doca_telemetry_schema *doca_schema, uint32_t timeout)

Set IPC socket timeout in milliseconds Timeout for IPC messaging socket. If timeout is reached during send_receive, the client is considered disconnected. Default value is 3000 milliseconds.

Parameters

doca_schema
Pointer to ipc timeout attribute.

timeout
Maximum socket timeout in milliseconds

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_ipc_sockets_dir (doca_telemetry_schema *doca_schema, const char *sockets_dir)

Set IPC socket directory. Default path is "/opt/mellanox/doca/services/telemetry/ipc_sockets".

Parameters

doca_schema
Pointer to DOCA schema.

sockets_dir
Path to a folder containing DOCA Telemetry Service (DTS) sockets.

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_schema_set_opaque_events_enabled (doca_telemetry_schema *doca_schema)

Enable opaque events Opaque events are disabled by default.

Parameters

doca_schema
Pointer to DOCA schema.

Description

Note:

Passing a doca_schema value of NULL will result in an undefined behavior.

doca_error_t doca_telemetry_schema_start (doca_telemetry_schema *doca_schema)

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

Parameters

doca_schema
Input schema to start.

Returns

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

- DOCA_ERROR_INITIALIZATION - in case of failure.

Description

Do NOT add new types after this function was called.

doca_error_t doca_telemetry_source_create (doca_telemetry_schema *doca_schema, doca_telemetry_source **doca_source)

Creates a single DOCA source from schema.

Parameters

doca_schema
Schema from which source will be created.
doca_source
pointer to DOCA source, or NULL on error.

Returns

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

- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

Description

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

doca_error_t doca_telemetry_source_destroy (doca_telemetry_source *doca_source)

Destructor for DOCA source.

Parameters

doca_source
Source to destroy.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.

doca_error_t doca_telemetry_source_flush (doca_telemetry_source *doca_source)

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

Parameters

doca_source
DOCA source to flush.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.

doca_error_t doca_telemetry_source_get_opaque_report_max_data_size (doca_telemetry_source *doca_source, uint32_t *max_data_size)

Get max data size for opaque report.

Parameters

doca_source
Source to report.
max_data_size
Maximal data size

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter(s).

doca_error_t doca_telemetry_source_opaque_report (doca_telemetry_source *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

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

- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

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_error_t doca_telemetry_source_report (doca_telemetry_source *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

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

- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

Description

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

__DOCA_EXPERIMENTAL void doca_telemetry_source_set_id (doca_telemetry_source *doca_source, const char *source_id)

Set source id.

Parameters

doca_source
Pointer to DOCA source.

source_id
Hostname or guid.

Description

Note:

Passing a doca_source value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void doca_telemetry_source_set_tag (doca_telemetry_source *doca_source, const char *source_tag)

Set source tag.

Parameters

doca_source
Pointer to DOCA source.

source_tag
User defined data-file name prefix.

Description

Note:

Passing a doca_source value of NULL will result in an undefined behavior.

doca_error_t doca_telemetry_source_start (doca_telemetry_source *doca_source)

Applies source attribute and starts DOCA source.

Parameters

doca_source
DOCA source to start.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if source attributes are not set.

- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

Description

Call this function to start reporting.

doca_error_t doca_telemetry_type_add_field (doca_telemetry_type *type, doca_telemetry_field *field)

Add DOCA telemetry field to type. The users loses the ownership of the field after a successful invocation of the function.

Parameters

type
Pointer to doca telemetry type.

field
DOCA Telemetry field to add.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry field.

Description

Note:

field should NOT be passed to another type after calling this function.

doca_error_t doca_telemetry_type_create (doca_telemetry_type **type)

Create new telemetry type.

Parameters

type
Pointer to the newly allocated type.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry field.

doca_error_t doca_telemetry_type_destroy (doca_telemetry_type *type)

Destroy doca telemetry type previously created by doca_telemetry_type_create().

Parameters

type
Pointer to type.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

Description

Note:

fields added to this type should NOT be used after calling this function.

#define DOCA_GUID_SIZE 16

DOCA GUID size.

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

NetFlow Application ID.

Note:

This GUID cannot change

#define DOCA_NETFLOW_DEFAULT_PORT 2055

NetFlow collector default port.

#define DOCA_TELEMETRY_FIELD_TYPE_BOOL "bool"

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

DOCA telemetry bool type

#define DOCA_TELEMETRY_FIELD_TYPE_CHAR "char"

DOCA telemetry char type.

#define DOCA_TELEMETRY_FIELD_TYPE_DOUBLE "double"

DOCA telemetry double type.

#define DOCA_TELEMETRY_FIELD_TYPE_FLOAT "float"

DOCA telemetry float type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT "int"

DOCA telemetry in type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT16 "int16_t"

DOCA telemetry int16 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT32 "int32_t"

DOCA telemetry int32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT64 "int64_t"

DOCA telemetry int64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT8 "int8_t"

DOCA telemetry int8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONG "long"

DOCA telemetry long type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONGLONG "long long"

DOCA telemetry longlong type.

#define DOCA_TELEMETRY_FIELD_TYPE_SHORT "short"

DOCA telemetry short type.

#define DOCA_TELEMETRY_FIELD_TYPE_TIMESTAMP DOCA_TELEMETRY_FIELD_TYPE_UINT64

DOCA telemetry timestamp type.

#define DOCA_TELEMETRY_FIELD_TYPE_UCHAR "unsigned char"

DOCA telemetry uchar type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT "unsigned int"

DOCA telemetry uint type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT16 "uint16_t"

DOCA telemetry uint16 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT32 "uint32_t"

DOCA telemetry uint32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT64 "uint64_t"

DOCA telemetry uint64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT8 "uint8_t"

DOCA telemetry uint8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONG "unsigned long"

DOCA telemetry ulong type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONGLONG "unsigned long long"

DOCA telemetry ulonglong type.

#define DOCA_TELEMETRY_FIELD_TYPE_USHORT "unsigned short"

DOCA telemetry ushort type.

2.29. Version Management

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

const char *doca_version (void)

Function returning DOCA's (SDK) version string.

Returns

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

Description

Note:

Represents the SDK version a project was compiled with.

const __DOCA_EXPERIMENTAL char *doca_version_runtime (void)

Function returning DOCA's (runtime) version string.

Returns

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

Description

Note:

Represents the runtime version a project is linked against.

#define DOCA_CURRENT_VERSION_NUM DOCA_VERSION_NUM(DOCA_VER_MAJOR, DOCA_VER_MINOR, DOCA_VER_PATCH)

Macro of current version number for comparisons.

#define DOCA_VER_MAJOR 2

Major version number 0-255.

#define DOCA_VER_MINOR 0

Minor version number 0-255.

#define DOCA_VER_PATCH 2027

Patch version number 0-9999.

#define DOCA_VER_STRING "2.0.2027"

DOCA Version String.

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

Check if the version specified is equal to current.

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

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

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

Macro of version number for comparisons.

2.3. 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.

2.11. 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.

2.12. 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.

2.14. 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 3. Data Structures

Here are the data structures with brief descriptions:

doca_compress_deflate_job

doca_compress_lz4_job

doca_ct_cfg
: Doca ct global configuration

doca_dma_job_memcpy

doca_dma_memcpy_result

doca_dpa_dev_event_remote_t

doca_dpi_job
: DOCA_DPI job definition

doca_dpi_parsing_info
: L2-L4 flow information, used to uniquely define a flow

doca_dpi_result
: DOCA_DPI result definition

doca_dpi_sig_data
: Extra signature data

doca_dpi_sig_info
: Signature info

doca_dpi_stat_info
: DPI statistics

doca_ec_job
: Galois multiplication. This job will galois multiply the src buffer with the coding matrix and output the result to dst buffer

doca_ec_job_create
: Jobs to be dispatched via EC library. This job will galois multiply the src buffer with the coding matrix and output the result to dst buffer

doca_ec_job_recover
: Recover job recover lost data blocks by using the remaining data blocks and redundancy blocks

doca_ec_job_update
: Update job Update redundancy blocks because a few data blocks were updated

doca_encryption_key
: IPSec encryption key

doca_event
Activity completion event
doca_flow_action_desc
Action description
doca_flow_action_descs
Action descriptions
doca_flow_action_descs_meta
Metadata action description per field
doca_flow_action_field
Extended modification action
doca_flow_actions
Doca flow actions information
doca_flow_aged_query
Aged flow query callback context
doca_flow_cfg
Doca flow global configuration
doca_flow_encap_action
Doca flow encap data information
doca_flow_fwd
Forwarding configuration
doca_flow_grpc_bindable_obj
Bindable object configuration
doca_flow_grpc_fwd
Forwarding configuration wrapper
doca_flow_grpc_pipe_cfg
Pipeline configuration wrapper
doca_flow_header_eth
Doca flow eth header
doca_flow_header_eth_vlan
Doca flow vlan header
doca_flow_header_format
Doca flow packet format
doca_flow_header_geneve
Doca flow GENEVE header
doca_flow_header_icmp
Doca flow icmp header in match data
doca_flow_header_ip4
Doca flow ipv4 header in match data
doca_flow_header_ip6
Doca flow ipv6 header in match data
doca_flow_header_l4_port
Doca flow tcp or udp port header in match data

doca_flow_header_mpls
Doca flow MPLS header
doca_flow_header_tcp
Doca flow tcp header in match data
doca_flow_header_udp
Doca flow udp header in match data
doca_flow_ip_addr
Doca flow ip address
doca_flow_match
Doca flow matcher information
doca_flow_meta
Doca flow meta data
doca_flow_mirror_target
Doca flow mirror target
doca_flow_monitor
Doca monitor action configuration
doca_flow_ordered_list
doca_flow_pipe_attr
Pipe attributes
doca_flow_pipe_cfg
Pipeline configuration
doca_flow_port_cfg
Doca flow port configuration
doca_flow_push_action
Doca flow push data information
doca_flow_query
Flow query result
doca_flow_resource_crypto_cfg
Doca flow crypto resource configuration
doca_flow_resource_meter_cfg
Doca flow meter resource configuration
doca_flow_resource_mirror_cfg
Doca flow mirror resource configuration
doca_flow_resource_rss_cfg
Doca flow rss resource configuration
doca_flow_resources
Doca flow resource quota
doca_flow_shared_resource_cfg
Doca flow shared resource configuration
doca_flow_shared_resource_result
Flow shared resources query result
doca_flow_tun
Doca flow tunnel information

doca_sync_event_result

3.1. doca_compress_deflate_job Struct Reference

DOCA COMPRESS Deflate job to be dispatched via COMPRESS library.

struct doca_job doca_compress_deflate_job::base

Common job data.

doca_buf *doca_compress_deflate_job::dst_buff

Destination data buffer.

uint64_t *doca_compress_deflate_job::output_chksum

Output checksum. If it is a compress job the checksum calculated is of the src_buf. If it is a decompress job the checksum result calculated is of the dst_buf. When the job processing will end, the output_chksum will contain the CRC checksum result in the lower 32bit and the Adler checksum result in the upper 32bit.

const doca_buf *doca_compress_deflate_job::src_buff

Source data buffer.

3.2. doca_compress_lz4_job Struct Reference

DOCA COMPRESS LZ4 job to be dispatched via COMPRESS library.

struct doca_job doca_compress_lz4_job::base

Common job data.

doca_buf *doca_compress_lz4_job::dst_buff

Destination data buffer.

uint64_t *doca_compress_lz4_job::output_chksum

Output checksum. If it is a compress job the checksum calculated is of the src_buf. If it is a decompress job the checksum result calculated is of the dst_buf. When the job processing will end, the output_chksum will contain the CRC checksum result in the lower 32bit and the Adler checksum result in the upper 32bit.

const doca_buf *doca_compress_lz4_job::src_buff

Source data buffer. The source buffer must be from local memory. Note: when using doca_buf linked list, the length of the first data element in the source buffer must be at least 4B.

3.3. doca_ct_cfg Struct Reference

doca ct global configuration

uint16_t doca_ct_cfg::aging_core

CT aging thread bind to CPU core.

uint32_t doca_ct_cfg::flags

CT behavior flags

void *doca_ct_cfg::ib_dev

IB verbs device context

void *doca_ct_cfg::ib_pd

device protection domain

uint32_t doca_ct_cfg::nb_arm_queues

number of ARM CT queues(thread).

uint32_t doca_ct_cfg::nb_arm_sessions

number of ARM CT sessions.

uint16_t doca_ct_cfg::tcp_session_del_s

time to kill TCP session after RST/FIN.

uint16_t doca_ct_cfg::tcp_timeout_s

TCP timeout in second.

uint16_t doca_ct_cfg::udp_timeout_s

UDP timeout in second.

3.4. doca_dma_job_memcpy Struct Reference

A job to be dispatched via the DMA library.

struct doca_job doca_dma_job_memcpy::base

Common job data

doca_buf *doca_dma_job_memcpy::dst_buff

Destination data buffer

const doca_buf *doca_dma_job_memcpy::src_buff

Source data buffer

3.5. doca_dma_memcpy_result Struct Reference

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

doca_error_t doca_dma_memcpy_result::result

Operation result

3.6. doca_dpa_dev_event_remote_t Struct Reference

DPA remote event handle type definition

uint64_t doca_dpa_dev_event_remote_t::data

Remote DPA event opaque

3.7. doca_dpi_job Struct Reference

DOCA_DPI job definition.

struct doca_job doca_dpi_job::base

Opaque struct.

doca_dpi_flow_ctx *doca_dpi_job::flow_ctx

The flow context handler, created by calling doca_dpi_flow_create().

bool doca_dpi_job::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).

uint32_t doca_dpi_job::payload_offset

Indicates where the packet's payload begins.

doca_buf *doca_dpi_job::pkt

The packet to be inspected.

doca_dpi_result *doca_dpi_job::result

The inspection result buffer, caller must pre-allocate it and ensure it is not freed until result is returned.

3.8. doca_dpi_parsing_info Struct Reference

L2-L4 flow information, used to uniquely define a flow.

doca_dpi_parsing_info::@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

3.9. doca_dpi_result Struct Reference

DOCA_DPI result definition.

struct doca_dpi_sig_info doca_dpi_result::info

Signature information

bool doca_dpi_result::matched

Indicates flow was matched

doca_buf *doca_dpi_result::pkt

The packet inspected for this result.

int doca_dpi_result::status_flags

doca_dpi_flow_status flags

3.10. 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

3.11. 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

3.12. 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

3.13. doca_ec_job Struct Reference

Galois multiplication. This job will galois multiply the src buffer with the coding matrix and output the result to dst buffer.

Note:

- recommended to work with other jobs for EC workflow
- src_buff and dst_buff should be in multiplication of block size. for example create job:

coding matrix is 10x4 - 10 original blocks, 4 redundancy blocks src_buff size : 10x64KB = 640KB rdnc_buff size : 4x64KB = 256KB

Note:

buff size should be at 64B padd. For example: 500B size should be padd to be 512B. minimum 64B.

struct doca_job doca_ec_job::base

Common job data.

doca_matrix *doca_ec_job::coding_matrix

coding matrix (see below doca_ec_matrix_from_raw)

doca_buf *doca_ec_job::dst_buff

Destination data buffer.

const doca_buf *doca_ec_job::src_buff

Source data buffer.

3.14. doca_ec_job_create Struct Reference

Jobs to be dispatched via EC library. This job will galois multiply the src buffer with the coding matrix and output the result to dst buffer.

Note:

src_buff and dst_buff should be in multiplication of block size. for example create job:

coding matrix is 10x4 - 10 original blocks, 4 redundancy blocks src_buff size : 10x64KB = 640KB rdnc_buff size : 4x64KB = 256KB

Note:

buff size should be at 64B padd. For example: 500B size should be padd to be 512B. minimum 64B.

create job: create redundancy blocks (backup blocks):

struct doca_job doca_ec_job_create::base

Common job data.

doca_matrix *doca_ec_job_create::create_matrix

create matrix (see below doca_ec_matrix_gen)

doca_buf *doca_ec_job_create::dst_rdnc_buff

Destination redundancy data buffer.

const doca_buf *doca_ec_job_create::src_original_data_buff

Source original data buffer.

3.15. doca_ec_job_recover Struct Reference

recover job recover lost data blocks by using the remaining data blocks and redundancy blocks

Note:

src_buff and dst_buff should be in multiplication of block size. for example create job:

coding matrix is 10x4 - 10 original blocks, 4 redundancy blocks src_buff size : 10x64KB = 640KB rdnc_buff size : 4x64KB = 256KB

Note:

buff size should be at 64B padd. For example: 500B size should be padd to be 512B. minimum 64B.

struct doca_job doca_ec_job_recover::base

Common job data.

doca_buf *doca_ec_job_recover::dst_recovered_data_buff

Destination data buffer.

doca_matrix *doca_ec_job_recover::recover_matrix

recover matrix (see below doca_ec_recover_matrix_create)

const doca_buf *doca_ec_job_recover::src_remaining_data_buff

Source remaining blocks buffer.

3.16. doca_ec_job_update Struct Reference

Update job Update redundancy blocks because a few data blocks were updated.

> Note:
>
> src_buff and dst_buff should be in multiplication of block size. for example create job:

coding matrix is 10x4 - 10 original blocks, 4 redundancy blocks src_buff size : 10x64KB = 640KB rdnc_buff size : 4x64KB = 256KB

> Note:
>
> buff size should be at 64B padd. For example: 500B size should be padd to be 512B. minimum 64B.

struct doca_job doca_ec_job_update::base

Common job data.

doca_buf *doca_ec_job_update::dst_updated_rdnc_buff

Destination updated redundancy blocks.

const doca_buf *doca_ec_job_update::src_data_rdnc_buff

Source old & new data with rdnc blocks.

doca_matrix *doca_ec_job_update::update_matrix

update matrix (see below doca_ec_update_matrix_gen)

3.17. doca_encryption_key Struct Reference

IPSec encryption key.

uint64_t doca_encryption_key::implicit_iv

The IV is inserted into the GCM engine is calculated by

void *doca_encryption_key::raw_key

Raw key buffer. Actual size of this buffer defined by type.

uint32_t doca_encryption_key::salt

The salt is inserted into the GCM engine is calculated by

enumdoca_encryption_key_type doca_encryption_key::type

size of enc key

3.18. doca_event Struct Reference

Activity completion event.

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

doca_data doca_event::result

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

int doca_event::type

The type of the event originating activity.

doca_data doca_event::user_data

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

3.19. doca_flow_action_desc Struct Reference

action description

enumdoca_flow_action_type doca_flow_action_desc::type

type

3.20. doca_flow_action_descs Struct Reference

action descriptions

struct doca_flow_action_desc doca_flow_action_descs::dscp_ecn

action description of DSCP/ECN.

struct doca_flow_action_desc doca_flow_action_descs::dst_ip

action description of destination IP.

struct doca_flow_action_desc doca_flow_action_descs::dst_mac

action description of destination MAC.

struct doca_flow_action_desc doca_flow_action_descs::dst_port

action description of destination L4 port.

struct doca_flow_action_desc doca_flow_action_descs::eth_type

action description of ether type.

struct doca_flow_action_descs_meta doca_flow_action_descs::meta

action description of meta data.

struct doca_flow_action_desc doca_flow_action_descs::src_ip

action description of source IP.

struct doca_flow_action_desc doca_flow_action_descs::src_mac

action description of source MAC.

struct doca_flow_action_desc doca_flow_action_descs::src_port

action description of source L4 port.

struct doca_flow_action_desc doca_flow_action_descs::ttl

action description of IPv4 TTL.

struct doca_flow_action_desc doca_flow_action_descs::tunnel

action description of tunnel.

struct doca_flow_action_desc doca_flow_action_descs::vlan_id

action description of VLAN ID.

3.21. doca_flow_action_descs_meta Struct Reference

Metadata action description per field.

struct doca_flow_action_desc doca_flow_action_descs_meta::hash

action description of hash. This field is read only, used by copy only as src.

struct doca_flow_action_desc doca_flow_action_descs_meta::pkt_meta

action description of pkt_meta.

struct doca_flow_action_desc doca_flow_action_descs_meta::u32

action description of meta.

3.22. doca_flow_action_field Struct Reference

extended modification action

void *doca_flow_action_field::address

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

uint32_t doca_flow_action_field::offset

If address is not NULL, bit offset within the field from the address. Otherwise, bit offset from the start of context field.

3.23. doca_flow_actions Struct Reference

doca flow actions information

uint8_t doca_flow_actions::action_idx

index according to place provided on creation

uint32_t doca_flow_actions::crypto_id

Crypto shared action id

bool doca_flow_actions::decap

when true, will do decap

struct doca_flow_encap_action doca_flow_actions::encap

encap data information

uint32_t doca_flow_actions::flags

action flags

bool doca_flow_actions::has_encap

when true, will do encap

bool doca_flow_actions::has_push

when true, push header

struct doca_flow_meta doca_flow_actions::meta

modify meta data, pipe action as mask

struct doca_flow_header_format doca_flow_actions::outer

modify outer headers

bool doca_flow_actions::pop

when true, pop header

enumdoca_flow_crypto_protocol_type doca_flow_actions::proto_type

Crypto shared action type

struct doca_flow_push_action doca_flow_actions::push

push header data information

doca_flow_actions::@12 doca_flow_actions::security

security shared action

struct doca_flow_tun doca_flow_actions::tun

modify tunnel headers

3.24. doca_flow_aged_query Struct Reference

aged flow query callback context

uint64_t doca_flow_aged_query::user_data

The user input context, otherwise the doca_flow_pipe_entry pointer

3.25. doca_flow_cfg Struct Reference

doca flow global configuration

doca_flow_entry_process_cb doca_flow_cfg::cb

callback for entry create/destroy

uint64_t doca_flow_cfg::flags

configuraton flags

const char *doca_flow_cfg::mode_args

set doca flow architecture mode switch, vnf

uint8_t doca_flow_cfg::nr_acl_collisions

number of collisions for the acl module

uint32_t doca_flow_cfg::nr_shared_resources

total shared resource per type

uint32_t doca_flow_cfg::queue_depth

Number of pre-configured queue_size, default to 128

uint16_t doca_flow_cfg::queues

queue id for each offload thread

struct doca_flow_resources doca_flow_cfg::resource

resource quota

doca_flow_shared_resource_unbind_cb doca_flow_cfg::unbind_cb

callback for unbinding of a shared resource

3.26. doca_flow_encap_action Struct Reference

doca flow encap data information

struct doca_flow_header_format doca_flow_encap_action::outer

outer header format

struct doca_flow_tun doca_flow_encap_action::tun

tunnel info

3.27. doca_flow_fwd Struct Reference

forwarding configuration

uint32_t doca_flow_fwd::idx

Index of the ordered list pipe entry.

doca_flow_pipe *doca_flow_fwd::next_pipe

next pipe pointer

int doca_flow_fwd::num_of_queues

number of queues

doca_flow_fwd::@13::@21 doca_flow_fwd::ordered_list_pipe

next ordered list pipe configuration

doca_flow_pipe *doca_flow_fwd::pipe

Ordered list pipe to select an entry from.

uint16_t doca_flow_fwd::port_id

destination port id

uint32_t doca_flow_fwd::rss_inner_flags

rss offload outer types

uint32_t doca_flow_fwd::rss_outer_flags

rss offload outer types

uint16_t *doca_flow_fwd::rss_queues

rss queues array

uint32_t doca_flow_fwd::shared_rss_id

shared rss id, only for pipe's fwd is NULL

doca_flow_target *doca_flow_fwd::target

pointer to target handler

enumdoca_flow_fwd_type doca_flow_fwd::type

indicate the forwarding type

3.28. doca_flow_grpc_bindable_obj Struct Reference

bindable object configuration

uint64_t doca_flow_grpc_bindable_obj::pipe_id

pipe id if type is pipe

uint32_t doca_flow_grpc_bindable_obj::port_id

port id if type is port

enumdoca_flow_grpc_bindable_obj_type doca_flow_grpc_bindable_obj::type

bindable object type

3.29. doca_flow_grpc_fwd Struct Reference

forwarding configuration wrapper

doca_flow_fwd *doca_flow_grpc_fwd::fwd

doca flow fwd struct

uint64_t doca_flow_grpc_fwd::next_pipe_id

next pipe id

3.30. doca_flow_grpc_pipe_cfg Struct Reference

pipeline configuration wrapper

doca_flow_pipe_cfg *doca_flow_grpc_pipe_cfg::cfg

doca_flow_pipe_cfg struct

uint16_t doca_flow_grpc_pipe_cfg::port_id

port id

3.31. doca_flow_header_eth Struct Reference

doca flow eth header

uint8_t doca_flow_header_eth::dst_mac

destination mac address

uint8_t doca_flow_header_eth::src_mac

source mac address

doca_be16_t doca_flow_header_eth::type

eth type

3.32. doca_flow_header_eth_vlan Struct Reference

doca flow vlan header

doca_be16_t doca_flow_header_eth_vlan::tci

vlan tci

3.33. doca_flow_header_format Struct Reference

doca flow packet format

struct doca_flow_header_eth doca_flow_header_format::eth

ether head

struct doca_flow_header_eth_vlan doca_flow_header_format::eth_vlan

vlan header array

struct doca_flow_header_icmp doca_flow_header_format::icmp

icmp header

struct doca_flow_header_ip4 doca_flow_header_format::ip4

ipv4 head

struct doca_flow_header_ip6 doca_flow_header_format::ip6

ipv6 head

uint16_t doca_flow_header_format::l2_valid_headers

indicate which headers are valid

enumdoca_flow_l3_type doca_flow_header_format::l3_type

layer 3 protocol type

enumdoca_flow_l4_type_ext doca_flow_header_format::l4_type_ext

l4 layer extend type

struct doca_flow_header_tcp doca_flow_header_format::tcp

tcp header

struct doca_flow_header_udp doca_flow_header_format::udp

udp header

3.34. doca_flow_header_geneve Struct Reference

doca flow GENEVE header.

doca_be16_t doca_flow_header_geneve::next_proto

next protocol

uint8_t doca_flow_header_geneve::o_c

OAM packet (1) + critical options present (1) + reserved (6).

uint8_t doca_flow_header_geneve::ver_opt_len

version (2) + options length (6).

doca_be32_t doca_flow_header_geneve::vni

geneve vni (24) + reserved (8).

3.35. doca_flow_header_icmp Struct Reference

doca flow icmp header in match data

uint8_t doca_flow_header_icmp::code

icmp code.

doca_be16_t doca_flow_header_icmp::ident

icmp identifier.

uint8_t doca_flow_header_icmp::type

icmp type

3.36. doca_flow_header_ip4 Struct Reference

doca flow ipv4 header in match data

uint8_t doca_flow_header_ip4::dscp_ecn

dscp and ecn

doca_be32_t doca_flow_header_ip4::dst_ip

ip dst address

uint8_t doca_flow_header_ip4::next_proto

ip next protocol

doca_be32_t doca_flow_header_ip4::src_ip

ip src address

uint8_t doca_flow_header_ip4::ttl

time to live

uint8_t doca_flow_header_ip4::version_ihl

version and internet header length

3.37. doca_flow_header_ip6 Struct Reference

doca flow ipv6 header in match data

uint8_t doca_flow_header_ip6::dscp_ecn

dscp and ecn

doca_be32_t doca_flow_header_ip6::dst_ip

ip dst address

uint8_t doca_flow_header_ip6::hop_limit

hop limit

uint8_t doca_flow_header_ip6::next_proto

ip next protocol

doca_be32_t doca_flow_header_ip6::src_ip

ip src address

3.38. doca_flow_header_l4_port Struct Reference

doca flow tcp or udp port header in match data

doca_be16_t doca_flow_header_l4_port::dst_port

destination port

doca_be16_t doca_flow_header_l4_port::src_port

source port

3.39. doca_flow_header_mpls Struct Reference

doca flow MPLS header.

0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-+ | Label | TC |S| TTL | +-+

doca_be32_t doca_flow_header_mpls::label

MPLS label.

3.40. doca_flow_header_tcp Struct Reference

doca flow tcp header in match data

uint8_t doca_flow_header_tcp::flags

tcp flags

struct doca_flow_header_l4_port doca_flow_header_tcp::l4_port

tcp source and destination port

3.41. doca_flow_header_udp Struct Reference

doca flow udp header in match data

struct doca_flow_header_l4_port doca_flow_header_udp::l4_port

udp source and destination port

3.42. 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

enumdoca_flow_l3_type doca_flow_ip_addr::type

ip address type

3.43. doca_flow_match Struct Reference

doca flow matcher information

uint32_t doca_flow_match::flags

match items which are no value

struct doca_flow_header_format doca_flow_match::inner

inner layer header format

struct doca_flow_meta doca_flow_match::meta

Programmable meta data.

struct doca_flow_header_format doca_flow_match::outer

outer layer header format

struct doca_flow_tun doca_flow_match::tun

tunnel info

3.44. doca_flow_meta Struct Reference

doca flow meta data

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

uint8_t doca_flow_meta::align

Structure alignment.

uint32_t doca_flow_meta::hairpin

Subject to forward using hairpin.

uint32_t doca_flow_meta::hash

Hash calculation to copy. Only used by modify field action, not matchable.

uint8_t doca_flow_meta::ipsec_syndrome

IPsec decrypt/authentication syndrome.

uint32_t doca_flow_meta::mark

Mark id.

uint8_t doca_flow_meta::nisp_syndrome

NISP decrypt/authentication syndrome.

uint32_t doca_flow_meta::pkt_meta

Shared with application via packet.

uint32_t doca_flow_meta::port_meta

Programmable source vport.

uint32_t doca_flow_meta::src

Source port in multi-port E-Switch mode

uint32_t doca_flow_meta::type

0: traffic 1: SYN 2: RST 3: FIN.

uint32_t doca_flow_meta::u32

Programmable user data.

uint32_t doca_flow_meta::zone

Zone ID for CT processing.

3.45. doca_flow_mirror_target Struct Reference

doca flow mirror target

struct doca_flow_encap_action doca_flow_mirror_target::encap

Encap data.

struct doca_flow_fwd doca_flow_mirror_target::fwd

Mirror target, must be filled.

bool doca_flow_mirror_target::has_encap

Encap mirrored packets.

3.46. doca_flow_monitor Struct Reference

doca monitor action configuration

uint32_t doca_flow_monitor::aging

aging time in seconds.

uint64_t doca_flow_monitor::cbs

Committed Burst Size (bytes).

uint64_t doca_flow_monitor::cir

Committed Information Rate (bytes/second).

uint8_t doca_flow_monitor::flags

indicate which actions be included

uint32_t doca_flow_monitor::shared_counter_id

shared counter id

uint32_t doca_flow_monitor::shared_meter_id

shared meter id

uint32_t doca_flow_monitor::shared_mirror_id

shared mirror id.

void *doca_flow_monitor::user_data

aging user data input.

3.47. doca_flow_ordered_list Struct Reference

Ordered list configuration.

const **doca_flow_ordered_list::elements

An array of DOCA flow structure pointers, depending on types.

uint32_t doca_flow_ordered_list::idx

List index among the lists of the pipe. At pipe creation, it must match the list position in the array of lists. At entry insertion, it determines which list to use.

uint32_t doca_flow_ordered_list::size

Number of elements in the list.

3.48. doca_flow_pipe_attr Struct Reference

pipe attributes

enumdoca_flow_pipe_domain doca_flow_pipe_attr::domain

pipe steering domain.

bool doca_flow_pipe_attr::is_root

pipeline is root or not. If true it means the pipe is a root pipe executed on packet arrival.

const char *doca_flow_pipe_attr::name

name for the pipeline

uint8_t doca_flow_pipe_attr::nb_actions

maximum number of doca flow action array, default is 1 if not set

uint32_t doca_flow_pipe_attr::nb_flows

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

uint8_t doca_flow_pipe_attr::nb_ordered_lists

number of ordered lists in the array, default 0, mutually exclusive with nb_actions

enumdoca_flow_pipe_type doca_flow_pipe_attr::type

type of pipe. enum doca_flow_pipe_type

3.49. doca_flow_pipe_cfg Struct Reference

pipeline configuration

**doca_flow_pipe_cfg::action_descs

action array descriptions

**doca_flow_pipe_cfg::actions

actions array for the pipeline

struct doca_flow_pipe_attr doca_flow_pipe_cfg::attr

attributes of pipe

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

**doca_flow_pipe_cfg::ordered_lists

array of ordered list types

doca_flow_port *doca_flow_pipe_cfg::port

port for the pipeline

3.50. 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

3.51. doca_flow_push_action Struct Reference

doca flow push data information

enumdoca_flow_push_action_type doca_flow_push_action::type

header type to push

3.52. 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

3.53. doca_flow_resource_crypto_cfg Struct Reference

doca flow crypto resource configuration

enumdoca_flow_crypto_action_type doca_flow_resource_crypto_cfg::action_type

crypto action

struct doca_flow_fwd doca_flow_resource_crypto_cfg::fwd

Crypto action continuation

enumdoca_flow_crypto_header_type doca_flow_resource_crypto_cfg::header_type

packet header type

uint8_t doca_flow_resource_crypto_cfg::key

Crypto key buffer

uint16_t doca_flow_resource_crypto_cfg::key_sz

key size in bytes

enumdoca_flow_crypto_net_type doca_flow_resource_crypto_cfg::net_type

packet network mode type

enumdoca_flow_crypto_protocol_type doca_flow_resource_crypto_cfg::proto_type

packet reformat action

uint8_t doca_flow_resource_crypto_cfg::reformat_data

reformat header buffer

uint16_t doca_flow_resource_crypto_cfg::reformat_data_sz

reformat header length in bytes

enumdoca_flow_crypto_icv_size doca_flow_resource_crypto_cfg::reformat_icv_sz

Intergtity Check Value in reformat action

enumdoca_flow_crypto_reformat_type doca_flow_resource_crypto_cfg::reformat_type

packet reformat action

void *doca_flow_resource_crypto_cfg::security_ctx

crypto object handle

3.54. doca_flow_resource_meter_cfg Struct Reference

doca flow meter resource configuration

uint64_t doca_flow_resource_meter_cfg::cbs

Committed Burst Size (bytes).

uint64_t doca_flow_resource_meter_cfg::cir

Committed Information Rate (bytes/second).

3.55. doca_flow_resource_mirror_cfg Struct Reference

doca flow mirror resource configuration

struct doca_flow_fwd doca_flow_resource_mirror_cfg::fwd

Original packet dst, can be filled optional.

int doca_flow_resource_mirror_cfg::nr_targets

Mirror target number.

doca_flow_mirror_target *doca_flow_resource_mirror_cfg::target

Mirror target pointer.

3.56. doca_flow_resource_rss_cfg Struct Reference

doca flow rss resource configuration

uint32_t doca_flow_resource_rss_cfg::inner_flags

rss offload inner types

int doca_flow_resource_rss_cfg::nr_queues

number of queues

uint32_t doca_flow_resource_rss_cfg::outer_flags

rss offload outer types

uint16_t *doca_flow_resource_rss_cfg::queues_array

rss queues array

3.57. doca_flow_resources Struct Reference

doca flow resource quota

uint32_t doca_flow_resources::nb_counters

Number of counters to configure

uint32_t doca_flow_resources::nb_meters

Number of traffic meters to configure

3.58. doca_flow_shared_resource_cfg Struct Reference

doca flow shared resource configuration

enumdoca_flow_pipe_domain doca_flow_shared_resource_cfg::domain

Shared resource steering domain

3.59. doca_flow_shared_resource_result Struct Reference

flow shared resources query result

3.60. doca_flow_tun Struct Reference

doca flow tunnel information

doca_be32_t doca_flow_tun::audp_hdr

Opaque audp tunnel header

doca_be32_t doca_flow_tun::esp_sn

ipsec sequence number

doca_be32_t doca_flow_tun::esp_spi

ipsec session parameter index

struct doca_flow_header_geneve doca_flow_tun::geneve

geneve header

doca_be32_t doca_flow_tun::gre_key

gre key

doca_be32_t doca_flow_tun::gtp_teid

gtp teid

bool doca_flow_tun::key_present

gre key is present

struct doca_flow_header_mpls doca_flow_tun::mpls

mpls labels

doca_be32_t doca_flow_tun::nisp_hdr

Opaque nisp tunnel header

doca_be16_t doca_flow_tun::protocol

next protocol

enumdoca_flow_tun_type doca_flow_tun::type

tunnel type

doca_be32_t doca_flow_tun::vxlan_tun_id

vxlan vni(24) + reserved (8).

3.61. doca_ipsec_sa_attr_egress Struct Reference

IPSec sa egress attributes - attributes for outgoing data.

uint32_t doca_ipsec_sa_attr_egress::sn_inc_enable

when set sn increment offloaded

3.62. doca_ipsec_sa_attr_ingress Struct Reference

IPSec sa egress attributes - attributes for incoming data.

uint32_t doca_ipsec_sa_attr_ingress::antireplay_enable

when enabled activates anti-replay protection window.

enumdoca_ipsec_replay_win_size doca_ipsec_sa_attr_ingress::replay_win_sz

Anti replay window size to enable sequence replay attack handling.

3.63. doca_ipsec_sa_attr_sn Struct Reference

IPSec sa sn attributes - attributes for sequence number - only if SN or AR enabled.

uint32_t doca_ipsec_sa_attr_sn::esn_enable

when set esn is enabled

uint32_t doca_ipsec_sa_attr_sn::esn_overlap

new/old indication of the High sequence number MSB - when set is old

uint64_t doca_ipsec_sa_attr_sn::sn_initial

set the initial sequence number

3.64. doca_ipsec_sa_attrs Struct Reference

IPSec attributes for create jobs.

enumdoca_ipsec_direction doca_ipsec_sa_attrs::direction

egress/ingress

struct doca_ipsec_sa_attr_egress doca_ipsec_sa_attrs::egress

< egress/ingress attr egress attr

struct doca_ipsec_sa_event_attrs doca_ipsec_sa_attrs::event

Reserve future use - ipsec events flags

enumdoca_ipsec_icv_length doca_ipsec_sa_attrs::icv_length

Authentication Tag length

struct doca_ipsec_sa_attr_ingress doca_ipsec_sa_attrs::ingress

ingress attr

struct doca_encryption_key doca_ipsec_sa_attrs::key

IPSec encryption key

struct doca_ipsec_sa_attr_sn doca_ipsec_sa_attrs::sn_attr

sn attributes

3.65. doca_ipsec_sa_create_job Struct Reference

DOCA IPSec SA creation job.

The result of this job if doca_workq_progress_retrieve returns:

- DOCA_SUCCESS - struct doca_event { .result.ptr } should point to new created `struct doca_ipsec_sa` object.
- DOCA_ERROR_IO_FAILED - struct doca_event { .result.u64 } should contain IPSec CTX specific error status code.

struct doca_job doca_ipsec_sa_create_job::base

doca job object

struct doca_ipsec_sa_attrs doca_ipsec_sa_create_job::sa_attrs

ipsec sa attr

3.66. doca_ipsec_sa_destroy_job Struct Reference

DOCA IPSec SA destroy job.

The result of this job as struct doca_event { .result.u64 } should contain SA destroy completion status code.

struct doca_job doca_ipsec_sa_destroy_job::base

doca job object

doca_ipsec_sa *doca_ipsec_sa_destroy_job::sa

ipsec sa object (from create)

3.67. doca_ipsec_sa_event_attrs Struct Reference

IPSec sa events attributes - when turned on will trigger an event.

uint32_t doca_ipsec_sa_event_attrs::esn_overlap_event_arm

1 when armed/to arm 0 otherwise.

uint32_t doca_ipsec_sa_event_attrs::hard_lifetime_arm

1 when armed/to arm 0 otherwise.

uint32_t doca_ipsec_sa_event_attrs::remove_flow_enable

1 when remove flow enabled/to enable; 0 otherwise.

uint32_t doca_ipsec_sa_event_attrs::remove_flow_packet_count

Packet counter, Decrements for every packet passing through the SA. Event are triggered occurs when the counter reaches soft- lifetime and hard-lifetime (0). When counter reaches hard-lifetime, all passing packets will return a relevant Syndrome.

uint32_t doca_ipsec_sa_event_attrs::remove_flow_soft_lifetime

Soft Lifetime threshold value. When remove_flow_packet_count reaches this value a soft lifetime event is triggered (if armed). See remove_flow_packet_count field in this struct fro more details.

uint32_t doca_ipsec_sa_event_attrs::soft_lifetime_arm

1 when armed/to arm 0 otherwise.

3.68. doca_job Struct Reference

Job structure describes request arguments for service provided by context.

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

DOCA Job layout

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

doca_ctx *doca_job::ctx

Doca CTX targeted by the job.

int doca_job::flags

Job submission flags (see `enum doca_job_flags`).

int doca_job::type

Defines the type of the job.

doca_data doca_job::user_data

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

3.69. doca_log_registrator

Registers log source on program start.
Logging Management cppClassifierVisibility: visibility=public cppClassifierTemplateModel: =

Should be used to register the log source. For example:

DOCA_LOG_REGISTER(dpi)

void foo { DOCA_LOG_INFO("Message"); }

Note:

The macro also takes care of the dtor() logic on teardown.

3.70. doca_pci_bdf Struct Reference

The PCI address of a device - same as the address in lspci.

3.71. doca_rdma_gid Struct Reference

gid struct

uint8_t doca_rdma_gid::raw

The raw value of the GID

3.72. doca_rdma_job_atomic Struct Reference

RDMA atomic job.

struct doca_job doca_rdma_job_atomic::base

Common job data

uint64_t doca_rdma_job_atomic::cmp_data

Value to compare for compare and swap.

doca_buf *doca_rdma_job_atomic::cmp_or_add_dest_buff

Destination data buffer

const doca_rdma_addr *doca_rdma_job_atomic::rdma_peer_addr

Optional: For RDMA context of type DC

doca_buf *doca_rdma_job_atomic::result_buff

Result of the atomic operation: remote original data before add, or remote original data before compare

uint64_t doca_rdma_job_atomic::swap_or_add_data

For add, the increment value for cmp, the new value to swap

3.73. doca_rdma_job_read_write Struct Reference

RDMA read/write job.

struct doca_job doca_rdma_job_read_write::base

Common job data

doca_buf *doca_rdma_job_read_write::dst_buff

Destination data buffer

doca_be32_t doca_rdma_job_read_write::immediate_data

Immediate data for write with imm

const doca_rdma_addr *doca_rdma_job_read_write::rdma_peer_addr

Optional: For RDMA context of type DC

const doca_buf *doca_rdma_job_read_write::src_buff

Source data buffer

3.74. doca_rdma_job_recv Struct Reference

The jobs to be dispatched via the RDMA library RDMA receive job.

struct doca_job doca_rdma_job_recv::base

Common job data

doca_buf *doca_rdma_job_recv::dst_buff

Destination data buffer, chain len must not exceed recv_buf_chain_len property

3.75. doca_rdma_job_send Struct Reference

RDMA send job.

struct doca_job doca_rdma_job_send::base

Common job data

doca_be32_t doca_rdma_job_send::immediate_data

Immediate data

const doca_rdma_addr *doca_rdma_job_send::rdma_peer_addr

Optional: For RDMA context of type UD or DC

const doca_buf *doca_rdma_job_send::src_buff

Source data buffer

3.76. doca_rdma_result Struct Reference

Result of a RDMA jobs. Will be held inside the doca_event::result field. RDMA Context results_fields_set() should be used to define which ones are required

doca_be32_t doca_rdma_result::immediate_data

Immediate data, valid only if opcode indicates

uint32_t doca_rdma_result::length

Length of transferred data in case of doca_rdma_job_recv or doca_rdma_job_read completion

enumdoca_rdma_opcode_t doca_rdma_result::opcode

Opcode in case of doca_rdma_job_recv completion

doca_rdma_addr *doca_rdma_result::rdma_peer_addr

Peer Address for UD and DC

doca_error_t doca_rdma_result::result

Operation result

3.77. doca_regex_job_search Struct Reference

Data required to dispatch a job to a RegEx engine.

uint8_t doca_regex_job_search::allow_batching

Set this to 1 to allow a RegEx device to choose to aggregate jobs into batches. Batching can improve throughput at the cost of latency. Set this to 0 to force this job to begin executing immediately, this will also force any previously enqueued jobs that have been batched and not yet dispatched to begin processing. Not all devices will support batching. If a device does not have batching support this flag is ignored.

struct doca_job doca_regex_job_search::base

Common job data.

const doca_buf *doca_regex_job_search::buffer

Data for the job.

doca_regex_search_result *doca_regex_job_search::result

Pointer to where the job response is stored. The caller must ensure this pointer is valid when submitting a job and it must remain valid until a response for the job has been retrieved from the RegEx engine. This object will be the returned via the event.result.ptr field.

uint16_t doca_regex_job_search::rule_group_ids

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

3.78. doca_regex_match Struct Reference

Description of a RegEx match

uint32_t doca_regex_match::length

Length of matched value.

uint32_t doca_regex_match::match_start

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

doca_regex_match *doca_regex_match::next

Allows matches to be linked together for easy management and iteration

uint32_t doca_regex_match::rule_id

ID of rule used to generate this match.

3.79. doca_regex_search_result Struct Reference

Result of a RegEx search

uint32_t doca_regex_search_result::detected_matches

Total number of detected matches.

doca_regex_match *doca_regex_search_result::matches

Returned matches. Contains num_matches elements as a linked list.

doca_regex_mempool *doca_regex_search_result::matches_mempool

Memory pool owning the matches.

uint32_t doca_regex_search_result::num_matches

Total number of returned matches.

uint64_t doca_regex_search_result::status_flags

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

3.80. doca_rmax_cpu_affinity_mask Struct Reference

Data structure to describe CPU mask for doca_rmax internal thread.

doca_rmax_cpu_mask_t doca_rmax_cpu_affinity_mask::cpu_bits

CPU is included in affinity mask if the corresponding bit is set

3.81. doca_rmax_in_stream_completion Struct Reference

Completion returned by input stream describing the incoming packets.

Input stream starts to receive packets right after start and attaching any flow.

uint32_t doca_rmax_in_stream_completion::elements_count

Number of packets received

**doca_rmax_in_stream_completion::memblk_ptr_arr

Array of pointers to the beginning of the memory block as configured by input stream create step. The offset between packets inside memory block can be queried by doca_rmax_in_stream_get_memblk_stride_size

uint32_t doca_rmax_in_stream_completion::memblk_ptr_arr_len

Number of memory blocks placed in memblk_ptr_arr. See doca_rmax_in_stream_get_memblks_count.

uint32_t doca_rmax_in_stream_completion::seqn_first

Sequnce number of the first packet

uint64_t doca_rmax_in_stream_completion::ts_first

Time of arrival of the first packet

uint64_t doca_rmax_in_stream_completion::ts_last

Time of arrival of the last packet

3.82. doca_rmax_job_rx_data Struct Reference

Receive data job.

struct doca_job doca_rmax_job_rx_data::base

Common job data

3.83. doca_rmax_stream_error Struct Reference

Detailed completion error information.

int doca_rmax_stream_error::code

Raw Rivermax error code

const char *doca_rmax_stream_error::message

Human-readable error

3.84. doca_sha_job Struct Reference

DOCA SHA job definition. -- "struct doca_sha_job" is used for one-shot SHA calculation. -- Its typical usage is: -- construct a job: struct doca_sha_job job = { .base.type = DOCA_SHA_JOB_SHA1, .req_buf = user_req_buf, .resp_buf = user_resp_buf, .flags = DOCA_SHA_JOB_FLAGS_NONE }; -- submit job: doca_workq_submit(workq, &job.base); -- retrieve event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);

-- For doca_workq_submit() return code: -- DOCA_SUCCESS: -- The job is submitted successfully. It also means: this submitted source data cannot be freely manipulated until its response is received. -- DOCA_ERROR_INVALID_VALUE: -- Some of the job attribute members use illegal value. for example, response buffer length is < 20bytes for SHA1; request buffer length == 0, and the job type attribute is not supported. -- DOCA_ERROR_NO_MEMORY: -- The job resouce is exhausted for now, we need to call progress_retrieve() first to receive response and free job resource, then call job_submit() to try again to submit the same job. -- DOCA_ERROR_BAD_STATE: -- sha_ctx is corrupted now, need reset.

-- For doca_workq_progress_retrieve() return code: -- DOCA_SUCCESS: -- we get a response from SHA engine. user can utilise doca_job's user_data field to setup special data to correlate the returned event and the corresponding job. -- DOCA_ERROR_AGAIN: -- In order to get a response, we need to call progress_retrieve() again. -- DOCA_ERROR_IO_FAILED: -- abnormal occurs in the SHA engine hardware queue, sha_ctx and workq need to be re-initialized. -- DOCA_ERROR_INVALID_VALUE: -- received invalid input.

struct doca_job doca_sha_job::base

Opaque structure.

uint64_t doca_sha_job::flags

SHA job flags. For the last segment of a doca_sha_partial_job, use DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL. Otherwise, use DOCA_SHA_JOB_FLAGS_NONE.

doca_buf *doca_sha_job::req_buf

User request. SHA engine accessible buffer pointed to the user input data. The req_buf can be a linked_list doca_buf, so that a chained multiple bufs can be used as valid request.

doca_buf *doca_sha_job::resp_buf

User response. The response byte count can be decided by DOCA_SHAXXX_BYTE_COUNT macro. The chained doca_buf is descouraged to be used as a response. Although resp_buf can be a linked_list doca_buf, no submission failure, but only the head element of the chained buf is used for now, because the SHA output is no more than 64bytes.

3.85. doca_sha_partial_job Struct Reference

DOCA SHA_PARTIAL job definition. -- "struct doca_sha_partial_job" is used for stateful SHA calculation. -- Its typical usage for a job composed of 3 segments is: -- get a session handle: doca_sha_partial_session *session; doca_sha_partial_session_create(ctx, workq, &session); -- construct the 1st job: struct doca_sha_partial_job job = { .sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_1st_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_NONE, .session = session, }; -- submit 1st segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 1st event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); The purpose of this call is to make sure the 1st_segment processing is finished before we can continue to send the next segment, because it is necessary to sequentially process all segment for generating correct SHA result. And the "user_resp_buf" at this moment contains garbage values. -- after the DOCA_SUCCESS event of the 1st segment is received, we can continue to

submit 2nd segment: -- construct the 2nd job: struct doca_sha_partial_job job = { .sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_2nd_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_NONE, .session = session, }; -- submit 2nd segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 2nd event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); The purpose of this call is also to make sure the 2nd_segment processing is finished. And the "user_resp_buf" at this moment still contains garbage values. -- after the DOCA_SUCCESS event of the 2nd segment is received, we can continue to submit 3rd/final segment: -- construct the 3rd job: struct doca_sha_partial_job job = { .sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_3rd_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL, .session = session, }; -- submit 3rd segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 3rd event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); -- After the DOCA_SUCCESS event of the 3rd segment is received, the whole job processing is done. We can get the expected SHA result from "user_resp_buf". -- release session: doca_sha_partial_session_destroy(ctx, workq, session); -- During the whole process, please make sure to use the same "session" handle. -- And for the last segment, the "DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL" flag must be set.

-- For doca_workq_submit() return code: -- DOCA_SUCCESS: -- The job is submitted successfully. It also means: this submitted source data cannot be freely manipulated until its response is received. -- DOCA_ERROR_INVALID_VALUE: -- Some of the job attribute members use illegal value. for example, response buffer length is < 20bytes for SHA1; request buffer length == 0, and the job type attribute is not supported. -- DOCA_ERROR_NO_MEMORY: -- The job resouce is exhausted for now, we need to call progress_retrieve() first to receive response and free job resource, then call job_submit() to try again to submit the same job. -- DOCA_ERROR_BAD_STATE: -- sha_ctx is corrupted now, need reset.

-- For doca_workq_progress_retrieve() return code: -- DOCA_SUCCESS: -- we get a response from SHA engine. user can utilise doca_job's user_data field to setup special data to correlate the returned event and the corresponding job. -- DOCA_ERROR_AGAIN: -- In order to get a response, we need to call progress_retrieve() again. -- DOCA_ERROR_IO_FAILED: -- abnormal occurs in the SHA engine hardware queue, sha_ctx and workq need to be re-initialized. -- DOCA_ERROR_INVALID_VALUE: -- received invalid input.

Note: -- sha_partial_job session requirement: -- make sure the same doca_sha_partial_session used for all segments of a whole job. -- before 1st segment submission, call doca_sha_partial_session_create() to grab a session handle. -- from the 1st to the last segment submission, always reuse the same session handle. -- after the last segment processing, to prevent a session resource leak, the user must explicitly call doca_sha_partial_session_destroy() to release this session handle. --

The doca_sha_partial_session_destroy() is provided to let user to free session handle at his will. -- If a session handle is released before the whole stateful SHA is finished, or if different handles are used for a stateful SHA, the job submission may fail due to job validity check failure; even the job submission successes, and the engine is not stalled, a wrong SHA result is expected. -- The "session" resource is limited, it is user's responsibility to make sure all allocated "session" handles are released. -- If "DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL" is not properly set, the engine will not be stalled, but a wrong SHA result is expected.

-- sha_partial_job segment length requirement: -- only the last segment allows seg-byte-count != multiple-of-64 for sha1 and sha256. For example, for the above example code, the 1st and 2nd segment byte length must be multiple of 64. -- only the last segment allows seg-byte-count != multiple-of-128 for sha512. -- If the above requirement is not met, job_submission will fail.

doca_sha_partial_session *doca_sha_partial_job::session

An opaque structure for user. Used to maintain state for stateful SHA calculation.

struct doca_sha_job doca_sha_partial_job::sha_job

A basic sha_job.

3.86. doca_sync_event_job_get Struct Reference

Get job to be dispatched on a DOCA WorkQ which attached to the Sync Event. Get the value of the Sync Event.

struct doca_job doca_sync_event_job_get::base

Inherits from DOCA Job

uint64_t *doca_sync_event_job_get::value

The Sync Event value

3.87. doca_sync_event_job_update_add Struct Reference

Add job to be dispatched on a DOCA WorkQ which attached to some Sync Event. Atomically increment the value of the Sync Event by the given value.

struct doca_job doca_sync_event_job_update_add::base

Inherits from DOCA Job

uint64_t *doca_sync_event_job_update_add::fetched

Fetched Sync Event value

uint64_t doca_sync_event_job_update_add::value

Value to set the Sync Event to

3.88. doca_sync_event_job_update_set Struct Reference

Set job to be dispatched on a DOCA WorkQ which attached to some Sync Event. Set the value of the Sync Event to the given value.

struct doca_job doca_sync_event_job_update_set::base

Inherits from DOCA Job

uint64_t doca_sync_event_job_update_set::value

Value to set the Sync Event to

3.89. doca_sync_event_job_wait Struct Reference

Wait job to be dispatched on a DOCA WorkQ which attached to some Sync Event. Wait for the Sync Event to be grater than the given value.

struct doca_job doca_sync_event_job_wait::base

Inherits from DOCA Job

uint64_t doca_sync_event_job_wait::mask

Mask for comparing the Sync Event value

uint64_t doca_sync_event_job_wait::value

Threshold to wait for the Sync Event to be grater than

3.90. doca_sync_event_result Struct Reference

Result of a Sync Event job. Held inside the doca_event::result.u64 field.

doca_error_t doca_sync_event_result::result

Operation status

Chapter 4. Data Fields

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

A

action
doca_dpi_sig_info
action_descs
doca_flow_pipe_cfg
action_idx
doca_flow_actions
action_type
doca_flow_resource_crypto_cfg
actions
doca_flow_pipe_cfg
address
doca_flow_action_field
aging
doca_flow_monitor
aging_core
doca_ct_cfg
align
doca_flow_meta
allow_batching
doca_regex_job_search
antireplay_enable
doca_ipsec_sa_attr_ingress
attr
doca_flow_pipe_cfg
audp_hdr
doca_flow_tun

B

base
- doca_compress_deflate_job
- doca_compress_lz4_job
- doca_sync_event_job_update_add
- doca_sync_event_job_update_set
- doca_sync_event_job_get
- doca_sync_event_job_wait
- doca_sha_job
- doca_dpi_job
- doca_regex_job_search
- doca_rdma_job_atomic
- doca_rdma_job_read_write
- doca_rdma_job_send
- doca_rdma_job_recv
- doca_ipsec_sa_destroy_job
- doca_ipsec_sa_create_job
- doca_ec_job_recover
- doca_ec_job_update
- doca_ec_job_create
- doca_ec_job
- doca_rmax_job_rx_data
- doca_dma_job_memcpy

buffer
- doca_regex_job_search

C

cb
- doca_flow_cfg

cbs
- doca_flow_resource_meter_cfg
- doca_flow_monitor

cfg
- doca_flow_grpc_pipe_cfg

cir
- doca_flow_resource_meter_cfg
- doca_flow_monitor

cmp_data
- doca_rdma_job_atomic

cmp_or_add_dest_buff
- doca_rdma_job_atomic

code
- doca_flow_header_icmp
- doca_rmax_stream_error

coding_matrix
- doca_ec_job

cpu_bits
- doca_rmax_cpu_affinity_mask

create_matrix
- doca_ec_job_create

crypto_id
- doca_flow_actions

ctx
- doca_job

D

data
- doca_dpa_dev_event_remote_t

decap
- doca_flow_actions

detected_matches
- doca_regex_search_result

devargs
- doca_flow_port_cfg

direction
- doca_ipsec_sa_attrs

domain
- doca_flow_shared_resource_cfg
- doca_flow_pipe_attr

dscp_ecn
- doca_flow_action_descs
- doca_flow_header_ip4
- doca_flow_header_ip6

dst_buff
- doca_rdma_job_read_write
- doca_compress_deflate_job
- doca_compress_lz4_job
- doca_rdma_job_recv
- doca_dma_job_memcpy
- doca_ec_job

dst_ip
- doca_flow_header_ip4
- doca_dpi_parsing_info
- doca_flow_action_descs

E

K

L

M

N

nb_arm_sessions
doca_ct_cfg
nb_counters
doca_flow_resources
nb_flows
doca_flow_pipe_attr
nb_http_parser_based
doca_dpi_stat_info
nb_matches
doca_dpi_stat_info
nb_meters
doca_flow_resources
nb_ordered_lists
doca_flow_pipe_attr
nb_other_l4
doca_dpi_stat_info
nb_other_l7
doca_dpi_stat_info
nb_scanned_pkts
doca_dpi_stat_info
nb_ssl_parser_based
doca_dpi_stat_info
nb_tcp_based
doca_dpi_stat_info
nb_udp_based
doca_dpi_stat_info
net_type
doca_flow_resource_crypto_cfg
next
doca_regex_match
next_pipe
doca_flow_fwd
next_pipe_id
doca_flow_grpc_fwd
next_proto
doca_flow_header_ip6
doca_flow_header_ip4
doca_flow_header_geneve
nisp_hdr
doca_flow_tun
nisp_syndrome
doca_flow_meta

O

P

Q

R

recover_matrix
- doca_ec_job_recover

reformat_data
- doca_flow_resource_crypto_cfg

reformat_data_sz
- doca_flow_resource_crypto_cfg

reformat_icv_sz
- doca_flow_resource_crypto_cfg

reformat_type
- doca_flow_resource_crypto_cfg

remove_flow_enable
- doca_ipsec_sa_event_attrs

remove_flow_packet_count
- doca_ipsec_sa_event_attrs

remove_flow_soft_lifetime
- doca_ipsec_sa_event_attrs

replay_win_sz
- doca_ipsec_sa_attr_ingress

req_buf
- doca_sha_job

resource
- doca_flow_cfg

resp_buf
- doca_sha_job

result
- doca_dpi_job
- doca_rdma_result
- doca_dma_memcpy_result
- doca_regex_job_search
- doca_sync_event_result
- doca_event

result_buff
- doca_rdma_job_atomic

rss_inner_flags
- doca_flow_fwd

rss_outer_flags
- doca_flow_fwd

rss_queues
- doca_flow_fwd

rule_group_ids
- doca_regex_job_search

rule_id
- doca_regex_match

S

T

U

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