



DOCA Libraries API

Reference Manual

Table of Contents

Chapter 1. Change Log.....	1
Chapter 2. Modules.....	2
2.1. DOCA AES-GCM Engine.....	3
doca_aes_gcm_key_type.....	3
doca_aes_gcm_task_decrypt_completion_cb_t.....	3
doca_aes_gcm_task_encrypt_completion_cb_t.....	3
doca_aes_gcm_as_ctx.....	4
doca_aes_gcm_cap_get_max_num_tasks.....	4
doca_aes_gcm_cap_task_decrypt_get_max_buf_size.....	5
doca_aes_gcm_cap_task_decrypt_get_max_iv_len.....	5
doca_aes_gcm_cap_task_decrypt_get_max_list_buf_num_elem.....	6
doca_aes_gcm_cap_task_decrypt_is_key_type_supported.....	7
doca_aes_gcm_cap_task_decrypt_is_supported.....	7
doca_aes_gcm_cap_task_decrypt_is_tag_128_supported.....	8
doca_aes_gcm_cap_task_decrypt_is_tag_96_supported.....	8
doca_aes_gcm_cap_task_encrypt_get_max_buf_size.....	9
doca_aes_gcm_cap_task_encrypt_get_max_iv_len.....	9
doca_aes_gcm_cap_task_encrypt_get_max_list_buf_num_elem.....	10
doca_aes_gcm_cap_task_encrypt_is_key_type_supported.....	11
doca_aes_gcm_cap_task_encrypt_is_supported.....	11
doca_aes_gcm_cap_task_encrypt_is_tag_128_supported.....	12
doca_aes_gcm_cap_task_encrypt_is_tag_96_supported.....	12
doca_aes_gcm_create.....	13
doca_aes_gcm_destroy.....	13
doca_aes_gcm_key_create.....	14
doca_aes_gcm_key_destroy.....	15
doca_aes_gcm_task_decrypt_alloc_init.....	15
doca_aes_gcm_task_decrypt_as_task.....	16
doca_aes_gcm_task_decrypt_get_aad_size.....	17
doca_aes_gcm_task_decrypt_get_dst.....	17
doca_aes_gcm_task_decrypt_get_iv.....	17
doca_aes_gcm_task_decrypt_get_key.....	18
doca_aes_gcm_task_decrypt_get_src.....	18
doca_aes_gcm_task_decrypt_get_tag_size.....	18
doca_aes_gcm_task_decrypt_set_aad_size.....	19

doca_aes_gcm_task_decrypt_set_conf.....	19
doca_aes_gcm_task_decrypt_set_dst.....	20
doca_aes_gcm_task_decrypt_set_iv.....	20
doca_aes_gcm_task_decrypt_set_key.....	21
doca_aes_gcm_task_decrypt_set_src.....	21
doca_aes_gcm_task_decrypt_set_tag_size.....	21
doca_aes_gcm_task_encrypt_alloc_init.....	22
doca_aes_gcm_task_encrypt_as_task.....	23
doca_aes_gcm_task_encrypt_get_aad_size.....	23
doca_aes_gcm_task_encrypt_get_dst.....	23
doca_aes_gcm_task_encrypt_get_iv.....	24
doca_aes_gcm_task_encrypt_get_key.....	24
doca_aes_gcm_task_encrypt_get_src.....	25
doca_aes_gcm_task_encrypt_get_tag_size.....	25
doca_aes_gcm_task_encrypt_set_aad_size.....	25
doca_aes_gcm_task_encrypt_set_conf.....	26
doca_aes_gcm_task_encrypt_set_dst.....	26
doca_aes_gcm_task_encrypt_set_iv.....	27
doca_aes_gcm_task_encrypt_set_key.....	27
doca_aes_gcm_task_encrypt_set_src.....	27
doca_aes_gcm_task_encrypt_set_tag_size.....	28
2.2. DOCA App Shield.....	28
DOCA App Shield Attributes.....	28
__doca_apsh_attst_info_get.....	28
__doca_apsh_container_info_get.....	29
__doca_apsh_envar_info_get.....	29
__doca_apsh_handle_info_get.....	30
__doca_apsh_injection_detect_info_get.....	30
__doca_apsh_ldrmodule_info_get.....	31
__doca_apsh_lib_info_get.....	31
__doca_apsh_module_info_get.....	32
__doca_apsh_netscan_info_get.....	32
__doca_apsh_privilege_info_get.....	33
__doca_apsh_process_info_get.....	33
__doca_apsh_process_parameters_info_get.....	34
__doca_apsh_sid_info_get.....	34
__doca_apsh_sys_config.....	35
__doca_apsh_thread_info_get.....	35

__doca_apsh_vad_info_get.....	36
__doca_apsh_yara_info_get.....	36
doca_apsh_attestation_free.....	37
doca_apsh_attestation_get.....	37
doca_apsh_attst_refresh.....	38
doca_apsh_container_processes_get.....	39
doca_apsh_containers_free.....	39
doca_apsh_containers_get.....	40
doca_apsh_create.....	40
doca_apsh_destroy.....	41
doca_apsh_dma_dev_set.....	41
doca_apsh_envars_free.....	42
doca_apsh_envars_get.....	42
doca_apsh_handles_free.....	43
doca_apsh_handles_get.....	43
doca_apsh_injection_detect_free.....	44
doca_apsh_injection_detect_get.....	44
doca_apsh_ldrmodules_free.....	45
doca_apsh_ldrmodules_get.....	45
doca_apsh_libs_free.....	46
doca_apsh_libs_get.....	46
doca_apsh_module_free.....	47
doca_apsh_modules_get.....	47
doca_apsh_netscan_free.....	48
doca_apsh_netscan_get.....	48
doca_apsh_privileges_free.....	49
doca_apsh_privileges_get.....	49
doca_apsh_process_netscan_get.....	50
doca_apsh_process_parameters_free.....	51
doca_apsh_process_parameters_get.....	52
doca_apsh_processes_free.....	53
doca_apsh_processes_get.....	53
doca_apsh_sids_free.....	54
doca_apsh_sids_get.....	54
doca_apsh_start.....	55
doca_apsh_sys_dev_set.....	55
doca_apsh_sys_kpgd_file_set.....	56
doca_apsh_sys_mem_region_set.....	56

doca_apsh_sys_os_symbol_map_set.....	57
doca_apsh_sys_os_type_set.....	58
doca_apsh_sys_set_scan_window_size.....	58
doca_apsh_sys_set_scan_window_step.....	59
doca_apsh_system_create.....	59
doca_apsh_system_destroy.....	60
doca_apsh_system_start.....	60
doca_apsh_threads_free.....	61
doca_apsh_threads_get.....	61
doca_apsh_vads_free.....	62
doca_apsh_vads_get.....	62
doca_apsh_yara_free.....	63
doca_apsh_yara_get.....	63
doca_apsh_attst_info_get.....	64
doca_apsh_container_info_get.....	64
doca_apsh_envar_info_get.....	65
doca_apsh_handle_info_get.....	65
doca_apsh_injection_detect_info_get.....	65
doca_apsh_ldrmodule_info_get.....	65
doca_apsh_lib_info_get.....	65
doca_apsh_module_info_get.....	66
doca_apsh_netscan_info_get.....	66
doca_apsh_privilege_info_get.....	66
doca_apsh_process_info_get.....	66
doca_apsh_process_parameters_info_get.....	67
doca_apsh_sid_info_get.....	67
doca_apsh_sys_config.....	67
doca_apsh_thread_info_get.....	67
doca_apsh_vad_info_get.....	67
doca_apsh_yara_info_get.....	68
2.2.1. DOCA App Shield Attributes.....	68
doca_apsh_attestation_attr.....	68
doca_apsh_container_attr.....	68
doca_apsh_envar_attr.....	69
doca_apsh_handle_attr.....	69
doca_apsh_injection_detect_attr.....	69
doca_apsh_ldrmodule_attr.....	70
doca_apsh_lib_attr.....	70

doca_apsh_module_attr.....	71
doca_apsh_netscan_attr.....	71
doca_apsh_privilege_attr.....	72
doca_apsh_process_attr.....	73
doca_apsh_process_parameters_attr.....	73
doca_apsh_sid_attr.....	74
doca_apsh_system_config_attr.....	74
doca_apsh_system_os.....	75
doca_apsh_thread_attr.....	75
doca_apsh_vad_attr.....	76
doca_apsh_yara_attr.....	76
doca_apsh_yara_rule.....	77
doca_apsh_yara_scan_type.....	77
DOCA_APSH_ATTESTATION_COMM_TYPE.....	77
DOCA_APSH_ATTESTATION_END_ADDRESS_TYPE.....	77
DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT_TYPE.....	77
DOCA_APSH_ATTESTATION_MATCHING_HASHES_TYPE.....	77
DOCA_APSH_ATTESTATION_PAGES_NUMBER_TYPE.....	77
DOCA_APSH_ATTESTATION_PAGES_PRESENT_TYPE.....	78
DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA_TYPE.....	78
DOCA_APSH_ATTESTATION_PID_TYPE.....	78
DOCA_APSH_ATTESTATION_PROTECTION_TYPE.....	78
DOCA_APSH_ATTESTATION_START_ADDRESS_TYPE.....	78
DOCA_APSH_CONTAINER_ID_TYPE.....	78
DOCA_APSH_DMA_DEV_TYPE.....	78
DOCA_APSH_ENVARS_PID_TYPE.....	78
DOCA_APSH_ENVARS_VALUE_TYPE.....	78
DOCA_APSH_ENVARS_VARIABLE_TYPE.....	78
DOCA_APSH_ENVARS_WINDOWS_BLOCK_TYPE.....	78
DOCA_APSH_HANDLE_ACCESS_TYPE.....	79
DOCA_APSH_HANDLE_NAME_TYPE.....	79
DOCA_APSH_HANDLE_PID_TYPE.....	79
DOCA_APSH_HANDLE_TABLE_ENTRY_TYPE.....	79
DOCA_APSH_HANDLE_TYPE_TYPE.....	79
DOCA_APSH_HANDLE_VALUE_TYPE.....	79
DOCA_APSH_HASHTEST_LIMIT_TYPE.....	79
DOCA_APSH_INJECTION_DETECT_PID_TYPE.....	79
DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_END_TYPE.....	79

DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_START_TYPE.....	79
DOCA_APSH_INJECTION_DETECT_VAD_END_TYPE.....	79
DOCA_APSH_INJECTION_DETECT_VAD_FILE_PATH_TYPE.....	80
DOCA_APSH_INJECTION_DETECT_VAD_PROTECTION_TYPE.....	80
DOCA_APSH_INJECTION_DETECT_VAD_START_TYPE.....	80
DOCA_APSH_INJECTION_DETECT_VAD_TAG_TYPE.....	80
DOCA_APSH_KPGD_FILE_TYPE.....	80
DOCA_APSH_LDRMODULE_BASE_ADDRESS_TYPE.....	80
DOCA_APSH_LDRMODULE_LIBRARY_PATH_TYPE.....	80
DOCA_APSH_LDRMODULE_PID_TYPE.....	80
DOCA_APSH_LDRMODULE_WINDOWS_DLL_NAME_TYPE.....	80
DOCA_APSH_LDRMODULE_WINDOWS_ININIT_TYPE.....	80
DOCA_APSH_LDRMODULE_WINDOWS_INLOAD_TYPE.....	81
DOCA_APSH_LDRMODULE_WINDOWS_INMEM_TYPE.....	81
DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE_TYPE.....	81
DOCA_APSH_LIB_LIBRARY_PATH_TYPE.....	81
DOCA_APSH_LIB_LINUX_LOAD_ADRESS_TYPE.....	81
DOCA_APSH_LIB_LOAD_ADRESS_TYPE.....	81
DOCA_APSH_LIB_PID_TYPE.....	81
DOCA_APSH_LIB_WINDOWS_DLL_NAME_TYPE.....	81
DOCA_APSH_LIB_WINDOWS_SIZE_OF_IMAGE_TYPE.....	81
DOCA_APSH_LIBS_LIMIT_TYPE.....	81
DOCA_APSH_MEM_REGION_TYPE.....	81
DOCA_APSH_MODULES_LIMIT_TYPE.....	82
DOCA_APSH_MODULES_NAME_TYPE.....	82
DOCA_APSH_MODULES_OFFSET_TYPE.....	82
DOCA_APSH_MODULES_SIZE_TYPE.....	82
DOCA_APSH_NETSCAN_COMM_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_FAMILY_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_FD_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_FILTER_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_NET_NAMESPACE_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_SOCKET_OFFSET_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_ACKED_TYPE.....	82
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_RECEIVED_TYPE.....	83
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_SENT_TYPE.....	83
DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_IN_TYPE.....	83
DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_OUT_TYPE.....	83

DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_IN_TYPE.....	83
DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_OUT_TYPE.....	83
DOCA_APSH_NETSCAN_LINUX_TYPE_TYPE.....	83
DOCA_APSH_NETSCAN_LOCAL_ADDR_TYPE.....	83
DOCA_APSH_NETSCAN_LOCAL_PORT_TYPE.....	83
DOCA_APSH_NETSCAN_PID_TYPE.....	83
DOCA_APSH_NETSCAN_PROTOCOL_TYPE.....	84
DOCA_APSH_NETSCAN_REMOTE_ADDR_TYPE.....	84
DOCA_APSH_NETSCAN_REMOTE_PORT_TYPE.....	84
DOCA_APSH_NETSCAN_STATE_TYPE.....	84
DOCA_APSH_NETSCAN_TIME_TYPE.....	84
DOCA_APSH_NETSCAN_WINDOWS_TIME_TYPE.....	84
DOCA_APSH_OS_SYMBOL_MAP_TYPE.....	84
DOCA_APSH_OS_TYPE_TYPE.....	84
DOCA_APSH_PRIVILEGES_IS_ON_TYPE.....	84
DOCA_APSH_PRIVILEGES_NAME_TYPE.....	84
DOCA_APSH_PRIVILEGES_PID_TYPE.....	84
DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT_TYPE.....	84
DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED_TYPE.....	85
DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT_TYPE.....	85
DOCA_APSH_PROCESS_COMM_TYPE.....	85
DOCA_APSH_PROCESS_CPU_TIME_TYPE.....	85
DOCA_APSH_PROCESS_LIMIT_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_GID_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_NS_MNT_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_NS_NET_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_NS_PID_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_STATE_TYPE.....	85
DOCA_APSH_PROCESS_LINUX_UID_TYPE.....	85
DOCA_APSH_PROCESS_PARAMETERS_CMD_LINE_TYPE.....	86
DOCA_APSH_PROCESS_PARAMETERS_IMAGE_BASE_ADDR_TYPE.....	86
DOCA_APSH_PROCESS_PARAMETERS_IMAGE_FULL_PATH_TYPE.....	86
DOCA_APSH_PROCESS_PARAMETERS_PID_TYPE.....	86
DOCA_APSH_PROCESS_PID_TYPE.....	86
DOCA_APSH_PROCESS_PPID_TYPE.....	86
DOCA_APSH_PROCESS_SID_ATTRIBUTES_TYPE.....	86
DOCA_APSH_PROCESS_SID_PID_TYPE.....	86
DOCA_APSH_PROCESS_SID_STRING_TYPE.....	86

DOCA_APSH_PROCESS_WINDOWS_EXIT_TIME_TYPE.....	86
DOCA_APSH_PROCESS_WINDOWS_OFFSET_TYPE.....	87
DOCA_APSH_PROCESS_WINDOWS_THREADS_TYPE.....	87
DOCA_APSH_SCAN_WIN_SIZE_TYPE.....	87
DOCA_APSH_SCAN_WIN_STEP_TYPE.....	87
DOCA_APSH_STRING_LIMIT_TYPE.....	87
DOCA_APSH_THREAD_LINUX_PROC_NAME_TYPE.....	87
DOCA_APSH_THREAD_LINUX_THREAD_NAME_TYPE.....	87
DOCA_APSH_THREAD_PID_TYPE.....	87
DOCA_APSH_THREAD_STATE_TYPE.....	87
DOCA_APSH_THREAD_TID_TYPE.....	87
DOCA_APSH_THREAD_WINDOWS_OFFSET_TYPE.....	87
DOCA_APSH_THREAD_WINDOWS_SUSPEND_COUNT_TYPE.....	88
DOCA_APSH_THREAD_WINDOWS_WAIT_REASON_TYPE.....	88
DOCA_APSH_THREADS_LIMIT_TYPE.....	88
DOCA_APSH_VADS_LIMIT_TYPE.....	88
DOCA_APSH_VHCA_ID_TYPE.....	88
DOCA_APSH_VMA_FILE_PATH_TYPE.....	88
DOCA_APSH_VMA_OFFSET_TYPE.....	88
DOCA_APSH_VMA_PID_TYPE.....	88
DOCA_APSH_VMA_PROCESS_NAME_TYPE.....	88
DOCA_APSH_VMA_PROTECTION_TYPE.....	88
DOCA_APSH_VMA_VM_END_TYPE.....	88
DOCA_APSH_VMA_VM_START_TYPE.....	88
DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE_TYPE.....	89
DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY_TYPE.....	89
DOCA_APSH_VMA_WINDOWS_TAG_TYPE.....	89
DOCA_APSH_WINDOWS_ENVARS_LIMIT_TYPE.....	89
DOCA_APSH_YARA_COMM_TYPE.....	89
DOCA_APSH_YARA_MATCH_WINDOW_ADDR_TYPE.....	89
DOCA_APSH_YARA_MATCH_WINDOW_LEN_TYPE.....	89
DOCA_APSH_YARA_PID_TYPE.....	89
DOCA_APSH_YARA_RULE_TYPE.....	89
2.3. DOCA Arg Parser.....	89
doca_argp_type.....	89
doca_argp_dpdk_cb_t.....	90
doca_argp_param_cb_t.....	90
doca_argp_validation_cb_t.....	90

doca_argp_destroy.....	90
doca_argp_get_log_level.....	91
doca_argp_get_sdk_log_level.....	91
doca_argp_init.....	91
doca_argp_param_create.....	92
doca_argp_param_destroy.....	92
doca_argp_param_set_arguments.....	93
doca_argp_param_set_callback.....	93
doca_argp_param_set_cli_only.....	94
doca_argp_param_set_description.....	94
doca_argp_param_set_long_name.....	95
doca_argp_param_set_mandatory.....	95
doca_argp_param_set_multiplicity.....	96
doca_argp_param_set_short_name.....	96
doca_argp_param_set_type.....	97
doca_argp_register_param.....	97
doca_argp_register_validation_callback.....	98
doca_argp_register_version_callback.....	98
doca_argp_set_dpdk_program.....	99
doca_argp_start.....	99
doca_argp_usage.....	100
2.4. DOCA Core.....	100
operations for DOCA Types.....	100
DOCA Buffer.....	100
DOCA Buffer Array.....	100
DOCA Buffer Inventory.....	100
DOCA Buffer Pool.....	100
DOCA Context.....	100
DOCA Device.....	101
2.4.1. operations for DOCA Types.....	101
__doca_builtin_ffsll.....	101
DOCA_BE16_GENMASK.....	101
DOCA_BE16_GET.....	101
DOCA_BE16_SET.....	101
DOCA_BE16P_GENMASK.....	102
DOCA_BE16P_SET.....	102
DOCA_BE32_GENMASK.....	102
DOCA_BE32_GET.....	102

DOCA_BE32_SET.....	102
DOCA_BE32P_GENMASK.....	103
DOCA_BE32P_SET.....	103
DOCA_BETOH16.....	103
DOCA_BETOH32.....	103
DOCA_HTOBE16.....	103
DOCA_HTOBE32.....	103
DOCA_SHIFT.....	103
DOCA_U8_GENMASK.....	103
DOCA_U8_GET.....	104
DOCA_U8_SET.....	104
DOCA_U8P_GENMASK.....	104
DOCA_U8P_SET.....	104
2.4.2. DOCA Buffer.....	104
doca_buf_chain_list.....	104
doca_buf_dec_refcount.....	105
doca_buf_get_data.....	106
doca_buf_get_data_len.....	106
doca_buf_get_head.....	106
doca_buf_get_last_in_list.....	107
doca_buf_get_len.....	107
doca_buf_get_list_len.....	107
doca_buf_get_next_in_list.....	108
doca_buf_get_refcount.....	108
doca_buf_inc_refcount.....	108
doca_buf_is_first_in_list.....	109
doca_buf_is_in_list.....	109
doca_buf_is_last_in_list.....	110
doca_buf_reset_data_len.....	110
doca_buf_set_data.....	111
doca_buf_set_data_len.....	111
doca_buf_unchain_list.....	112
2.4.3. DOCA Buffer Array.....	113
doca_dpa_dev_buf_arr_t.....	113
doca_buf_arr_create.....	113
doca_buf_arr_destroy.....	114
doca_buf_arr_get_dpa_handle.....	114
doca_buf_arr_get_gpu_handle.....	115

doca_buf_arr_set_params.....	115
doca_buf_arr_set_target_dpa.....	116
doca_buf_arr_set_target_gpu.....	116
doca_buf_arr_start.....	117
doca_buf_arr_stop.....	117
2.4.4. DOCA Buffer Inventory.....	118
doca_buf_inventory_buf_dup.....	118
doca_buf_inventory_buf_get_by_addr.....	119
doca_buf_inventory_buf_get_by_args.....	119
doca_buf_inventory_buf_get_by_data.....	120
doca_buf_inventory_create.....	121
doca_buf_inventory_destroy.....	121
doca_buf_inventory_expand.....	122
doca_buf_inventory_get_num_elements.....	123
doca_buf_inventory_get_num_free_elements.....	123
doca_buf_inventory_get_user_data.....	124
doca_buf_inventory_set_user_data.....	124
doca_buf_inventory_start.....	125
doca_buf_inventory_stop.....	125
2.4.5. DOCA Buffer Pool.....	126
doca_buf_pool_buf_alloc.....	126
doca_buf_pool_create.....	127
doca_buf_pool_destroy.....	127
doca_buf_pool_get_element_alignment.....	128
doca_buf_pool_get_num_elements.....	129
doca_buf_pool_get_num_free_elements.....	129
doca_buf_pool_get_user_data.....	129
doca_buf_pool_set_element_alignment.....	130
doca_buf_pool_set_user_data.....	130
doca_buf_pool_start.....	131
doca_buf_pool_stop.....	131
2.4.6. DOCA Context.....	132
doca_ctx_states.....	132
doca_ctx_state_changed_callback_t.....	133
doca_ctx_flush_tasks.....	133
doca_ctx_get_num_inflight_tasks.....	134
doca_ctx_get_state.....	134
doca_ctx_get_user_data.....	135

doca_ctx_set_datapath_on_dpa.....	135
doca_ctx_set_datapath_on_gpu.....	136
doca_ctx_set_state_changed_cb.....	136
doca_ctx_set_user_data.....	137
doca_ctx_start.....	137
doca_ctx_stop.....	138
2.4.7. DOCA Device.....	138
doca_devinfo_rep_filter.....	139
doca_dev_as_devinfo.....	139
doca_dev_close.....	139
doca_dev_open.....	140
doca_dev_rep_as_devinfo.....	140
doca_dev_rep_close.....	141
doca_dev_rep_open.....	141
doca_devinfo_cap_is_hotplug_manager_supported.....	142
doca_devinfo_cap_is_notification_moderation_supported.....	142
doca_devinfo_create_list.....	143
doca_devinfo_destroy_list.....	143
doca_devinfo_get_active_rate.....	144
doca_devinfo_get_ibdev_name.....	144
doca_devinfo_get_iface_name.....	145
doca_devinfo_get_ipv4_addr.....	146
doca_devinfo_get_ipv6_addr.....	146
doca_devinfo_get_lid.....	147
doca_devinfo_get_mac_addr.....	147
doca_devinfo_get_pci_addr_str.....	148
doca_devinfo_is_equal_pci_addr.....	148
doca_devinfo_rep_cap_is_filter_all_supported.....	149
doca_devinfo_rep_cap_is_filter_emulated_supported.....	150
doca_devinfo_rep_cap_is_filter_net_supported.....	150
doca_devinfo_rep_create_list.....	151
doca_devinfo_rep_destroy_list.....	152
doca_devinfo_rep_get_is_hotplug.....	152
doca_devinfo_rep_get_pci_addr_str.....	153
doca_devinfo_rep_get_pci_func_type.....	154
doca_devinfo_rep_get_vuid.....	154
doca_devinfo_rep_is_equal_pci_addr.....	155
DOCA_DEVINFO_IBDEV_NAME_SIZE.....	155

DOCA_DEVINFO_IFACE_NAME_SIZE.....	156
DOCA_DEVINFO_IPV4_ADDR_SIZE.....	156
DOCA_DEVINFO_IPV6_ADDR_SIZE.....	156
DOCA_DEVINFO_MAC_ADDR_SIZE.....	156
DOCA_DEVINFO_PCI_ADDR_SIZE.....	156
DOCA_DEVINFO_PCI_BDF_SIZE.....	156
DOCA_DEVINFO_REP_PCI_ADDR_SIZE.....	156
DOCA_DEVINFO_REP_PCI_BDF_SIZE.....	156
DOCA_DEVINFO_REP_VUID_SIZE.....	156
DOCA_DEVINFO_VUID_SIZE.....	156
2.5. DOCA Comch.....	156
DOCA Comch Consumer.....	156
DOCA Comch MsgQ.....	157
DOCA Comch Producer.....	157
doca_comch_counter.....	157
doca_comch_event_connection_status_changed_cb_t.....	157
doca_comch_event_consumer_cb_t.....	157
doca_comch_event_msg_rcv_cb_t.....	158
doca_comch_task_send_completion_cb_t.....	158
doca_comch_cap_client_is_supported.....	158
doca_comch_cap_get_max_clients.....	159
doca_comch_cap_get_max_msg_size.....	159
doca_comch_cap_get_max_name_len.....	160
doca_comch_cap_get_max_rcv_queue_size.....	160
doca_comch_cap_get_max_send_tasks.....	161
doca_comch_cap_server_is_supported.....	161
doca_comch_client_as_ctx.....	162
doca_comch_client_create.....	162
doca_comch_client_destroy.....	163
doca_comch_client_event_consumer_register.....	163
doca_comch_client_event_msg_rcv_register.....	164
doca_comch_client_get_client_ctx.....	164
doca_comch_client_get_connection.....	165
doca_comch_client_get_device.....	165
doca_comch_client_get_max_msg_size.....	166
doca_comch_client_get_rcv_queue_size.....	166
doca_comch_client_set_max_msg_size.....	167
doca_comch_client_set_rcv_queue_size.....	167

doca_comch_client_task_send_alloc_init.....	168
doca_comch_client_task_send_set_conf.....	169
doca_comch_connection_get_counter.....	170
doca_comch_connection_get_user_data.....	170
doca_comch_connection_set_user_data.....	171
doca_comch_connection_update_info.....	171
doca_comch_server_as_ctx.....	172
doca_comch_server_create.....	172
doca_comch_server_destroy.....	173
doca_comch_server_disconnect.....	173
doca_comch_server_event_connection_status_changed_register.....	174
doca_comch_server_event_consumer_register.....	175
doca_comch_server_event_msg_rcv_register.....	175
doca_comch_server_get_device.....	176
doca_comch_server_get_device_rep.....	176
doca_comch_server_get_max_msg_size.....	177
doca_comch_server_get_rcv_queue_size.....	177
doca_comch_server_get_server_ctx.....	178
doca_comch_server_set_max_msg_size.....	178
doca_comch_server_set_rcv_queue_size.....	179
doca_comch_server_task_send_alloc_init.....	179
doca_comch_server_task_send_set_conf.....	180
doca_comch_task_send_as_task.....	181
2.5.1. DOCA Comch Consumer.....	181
doca_comch_consumer_task_post_rcv_completion_cb_t.....	181
doca_dpa_dev_comch_consumer_completion_t.....	182
doca_dpa_dev_comch_consumer_t.....	182
doca_comch_consumer_as_ctx.....	182
doca_comch_consumer_cap_get_max_buf_list_len.....	182
doca_comch_consumer_cap_get_max_buf_size.....	183
doca_comch_consumer_cap_get_max_consumers.....	183
doca_comch_consumer_cap_get_max_imm_data_len.....	184
doca_comch_consumer_cap_get_max_num_tasks.....	184
doca_comch_consumer_cap_is_supported.....	185
doca_comch_consumer_completion_create.....	185
doca_comch_consumer_completion_destroy.....	185
doca_comch_consumer_completion_get_dpa_handle.....	186
doca_comch_consumer_completion_get_imm_data_len.....	186

doca_comch_consumer_completion_get_max_num_consumers.....	187
doca_comch_consumer_completion_get_max_num_rcv.....	187
doca_comch_consumer_completion_set_dpa_thread.....	188
doca_comch_consumer_completion_set_imm_data_len.....	188
doca_comch_consumer_completion_set_max_num_consumers.....	189
doca_comch_consumer_completion_set_max_num_rcv.....	189
doca_comch_consumer_completion_start.....	190
doca_comch_consumer_completion_stop.....	190
doca_comch_consumer_create.....	191
doca_comch_consumer_destroy.....	191
doca_comch_consumer_get_dpa_handle.....	192
doca_comch_consumer_get_id.....	192
doca_comch_consumer_get_imm_data_len.....	193
doca_comch_consumer_set_completion.....	193
doca_comch_consumer_set_dev_max_num_rcv.....	194
doca_comch_consumer_set_imm_data_len.....	194
doca_comch_consumer_task_post_rcv_alloc_init.....	195
doca_comch_consumer_task_post_rcv_as_task.....	196
doca_comch_consumer_task_post_rcv_get_buf.....	196
doca_comch_consumer_task_post_rcv_get_imm_data.....	196
doca_comch_consumer_task_post_rcv_get_imm_data_len.....	197
doca_comch_consumer_task_post_rcv_get_producer_id.....	197
doca_comch_consumer_task_post_rcv_set_buf.....	198
doca_comch_consumer_task_post_rcv_set_conf.....	198
2.5.2. DOCA Comch MsgQ.....	199
doca_comch_msgq_consumer_create.....	199
doca_comch_msgq_create.....	200
doca_comch_msgq_destroy.....	200
doca_comch_msgq_producer_create.....	200
doca_comch_msgq_set_dpa_consumer.....	201
doca_comch_msgq_set_dpa_producer.....	201
doca_comch_msgq_set_max_num_consumers.....	202
doca_comch_msgq_set_max_num_producers.....	202
doca_comch_msgq_start.....	202
doca_comch_msgq_stop.....	203
2.5.3. DOCA Comch Producer.....	203
doca_comch_producer_task_send_completion_cb_t.....	204
doca_dpa_dev_comch_producer_t.....	204

doca_comch_producer_as_ctx.....	204
doca_comch_producer_cap_get_max_buf_list_len.....	204
doca_comch_producer_cap_get_max_buf_size.....	205
doca_comch_producer_cap_get_max_num_tasks.....	205
doca_comch_producer_cap_get_max_producers.....	206
doca_comch_producer_cap_is_supported.....	206
doca_comch_producer_create.....	207
doca_comch_producer_destroy.....	207
doca_comch_producer_dpa_completion_attach.....	208
doca_comch_producer_get_dpa_handle.....	208
doca_comch_producer_get_id.....	209
doca_comch_producer_set_dev_max_num_send.....	209
doca_comch_producer_task_send_alloc_init.....	210
doca_comch_producer_task_send_as_task.....	210
doca_comch_producer_task_send_get_buf.....	211
doca_comch_producer_task_send_get_consumer_id.....	211
doca_comch_producer_task_send_get_imm_data.....	212
doca_comch_producer_task_send_get_imm_data_len.....	212
doca_comch_producer_task_send_set_buf.....	212
doca_comch_producer_task_send_set_conf.....	213
doca_comch_producer_task_send_set_consumer_id.....	213
doca_comch_producer_task_send_set_imm_data.....	214
2.6. DOCA Comm Channel.....	214
doca_comm_channel_msg_flags.....	214
doca_event_channel_t.....	215
doca_comm_channel_ep_connect.....	215
doca_comm_channel_ep_create.....	216
doca_comm_channel_ep_destroy.....	216
doca_comm_channel_ep_disconnect.....	217
doca_comm_channel_ep_event_handle_arm_recv.....	217
doca_comm_channel_ep_event_handle_arm_send.....	218
doca_comm_channel_ep_get_device.....	218
doca_comm_channel_ep_get_device_rep.....	219
doca_comm_channel_ep_get_event_channel.....	219
doca_comm_channel_ep_get_max_msg_size.....	220
doca_comm_channel_ep_get_peer_addr_list.....	220
doca_comm_channel_ep_get_pending_connections.....	221
doca_comm_channel_ep_get_recv_queue_size.....	222

doca_comm_channel_ep_get_send_queue_size.....	222
doca_comm_channel_ep_get_service_event_channel.....	223
doca_comm_channel_ep_listen.....	224
doca_comm_channel_ep_recvfrom.....	225
doca_comm_channel_ep_sendto.....	226
doca_comm_channel_ep_set_device.....	226
doca_comm_channel_ep_set_device_rep.....	227
doca_comm_channel_ep_set_max_msg_size.....	227
doca_comm_channel_ep_set_rcv_queue_size.....	228
doca_comm_channel_ep_set_send_queue_size.....	228
doca_comm_channel_ep_update_service_state_info.....	229
doca_comm_channel_get_max_message_size.....	230
doca_comm_channel_get_max_rcv_queue_size.....	230
doca_comm_channel_get_max_send_queue_size.....	231
doca_comm_channel_get_max_service_name_len.....	231
doca_comm_channel_get_service_max_num_connections.....	232
doca_comm_channel_peer_addr_get_rcv_bytes.....	232
doca_comm_channel_peer_addr_get_rcv_messages.....	233
doca_comm_channel_peer_addr_get_send_bytes.....	234
doca_comm_channel_peer_addr_get_send_in_flight_messages.....	234
doca_comm_channel_peer_addr_get_send_messages.....	235
doca_comm_channel_peer_addr_get_user_data.....	235
doca_comm_channel_peer_addr_set_user_data.....	236
doca_comm_channel_peer_addr_update_info.....	236
2.7. DOCA Compatibility Management.....	237
__attribute__.....	237
DLL_EXPORT_ATTR.....	237
DOCA_DEPRECATED.....	237
DOCA_EXPERIMENTAL.....	238
DOCA_STABLE.....	238
DOCA_STRUCT_START.....	238
2.8. DOCA Compress Engine.....	238
doca_compress_task_compress_deflate_completion_cb_t.....	238
doca_compress_task_decompress_deflate_completion_cb_t.....	238
doca_compress_task_decompress_lz4_block_completion_cb_t.....	239
doca_compress_task_decompress_lz4_stream_completion_cb_t.....	239
doca_compress_as_ctx.....	239
doca_compress_cap_get_max_num_tasks.....	240

doca_compress_cap_task_compress_deflate_get_max_buf_list_len.....	240
doca_compress_cap_task_compress_deflate_get_max_buf_size.....	241
doca_compress_cap_task_compress_deflate_is_supported.....	242
doca_compress_cap_task_decompress_deflate_get_max_buf_list_len.....	242
doca_compress_cap_task_decompress_deflate_get_max_buf_size.....	243
doca_compress_cap_task_decompress_deflate_is_supported.....	243
doca_compress_cap_task_decompress_lz4_block_get_max_buf_list_len.....	244
doca_compress_cap_task_decompress_lz4_block_get_max_buf_size.....	245
doca_compress_cap_task_decompress_lz4_block_is_supported.....	245
doca_compress_cap_task_decompress_lz4_stream_get_max_buf_list_len.....	246
doca_compress_cap_task_decompress_lz4_stream_get_max_buf_size.....	246
doca_compress_cap_task_decompress_lz4_stream_is_supported.....	247
doca_compress_create.....	247
doca_compress_destroy.....	248
doca_compress_task_compress_deflate_alloc_init.....	248
doca_compress_task_compress_deflate_as_task.....	249
doca_compress_task_compress_deflate_get_adler_cs.....	249
doca_compress_task_compress_deflate_get_crc_cs.....	250
doca_compress_task_compress_deflate_get_dst.....	250
doca_compress_task_compress_deflate_get_src.....	251
doca_compress_task_compress_deflate_set_conf.....	251
doca_compress_task_compress_deflate_set_dst.....	252
doca_compress_task_compress_deflate_set_src.....	252
doca_compress_task_decompress_deflate_alloc_init.....	252
doca_compress_task_decompress_deflate_as_task.....	253
doca_compress_task_decompress_deflate_get_adler_cs.....	254
doca_compress_task_decompress_deflate_get_crc_cs.....	254
doca_compress_task_decompress_deflate_get_dst.....	255
doca_compress_task_decompress_deflate_get_src.....	255
doca_compress_task_decompress_deflate_set_conf.....	256
doca_compress_task_decompress_deflate_set_dst.....	256
doca_compress_task_decompress_deflate_set_src.....	257
doca_compress_task_decompress_lz4_block_alloc_init.....	257
doca_compress_task_decompress_lz4_block_as_task.....	258
doca_compress_task_decompress_lz4_block_get_crc_cs.....	258
doca_compress_task_decompress_lz4_block_get_dst.....	259
doca_compress_task_decompress_lz4_block_get_src.....	259
doca_compress_task_decompress_lz4_block_get_xxh_cs.....	259

doca_compress_task_decompress_lz4_block_set_conf.....	260
doca_compress_task_decompress_lz4_block_set_dst.....	261
doca_compress_task_decompress_lz4_block_set_src.....	261
doca_compress_task_decompress_lz4_stream_alloc_init.....	262
doca_compress_task_decompress_lz4_stream_as_task.....	263
doca_compress_task_decompress_lz4_stream_get_are_blocks_independent.....	263
doca_compress_task_decompress_lz4_stream_get_crc_cs.....	263
doca_compress_task_decompress_lz4_stream_get_dst.....	264
doca_compress_task_decompress_lz4_stream_get_has_block_checksum.....	264
doca_compress_task_decompress_lz4_stream_get_src.....	265
doca_compress_task_decompress_lz4_stream_get_xlh_cs.....	265
doca_compress_task_decompress_lz4_stream_set_are_blocks_independent.....	266
doca_compress_task_decompress_lz4_stream_set_conf.....	266
doca_compress_task_decompress_lz4_stream_set_dst.....	267
doca_compress_task_decompress_lz4_stream_set_has_block_checksum.....	267
doca_compress_task_decompress_lz4_stream_set_src.....	267
2.9. DOCA Environment Configurations.....	268
DOCA_COMPAT_HELPERS.....	268
2.10. DOCA Device Emulation.....	268
DOCA Device Emulation - PCI Devices.....	268
DOCA Device Emulation - Virtio FS Devices.....	268
DOCA Device Emulation - Virtio Devices.....	268
2.10.1. DOCA Device Emulation - PCI Devices.....	268
DOCA Device Emulation - PCI Device Types.....	268
doca_devemu_pci_hotplug_state.....	268
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_handler_cb_t.....	269
doca_devemu_pci_dev_event_flr_handler_cb_t.....	269
doca_devemu_pci_dev_event_hotplug_state_change_handler_cb_t.....	270
doca_dpa_dev_devemu_pci_db_completion_t.....	270
doca_dpa_dev_devemu_pci_db_t.....	270
doca_dpa_dev_devemu_pci_msix_t.....	270
doca_devemu_pci_cap_get_max_hotplug_devices.....	270
doca_devemu_pci_db_completion_create.....	271
doca_devemu_pci_db_completion_destroy.....	271
doca_devemu_pci_db_completion_get_curr_num_dbs.....	272
doca_devemu_pci_db_completion_get_dpa_handle.....	272
doca_devemu_pci_db_completion_get_max_num_dbs.....	273
doca_devemu_pci_db_completion_set_max_num_dbs.....	273

doca_devemu_pci_db_completion_start.....	273
doca_devemu_pci_db_completion_stop.....	274
doca_devemu_pci_db_create_on_dpa.....	275
doca_devemu_pci_db_destroy.....	276
doca_devemu_pci_db_get_dpa_handle.....	276
doca_devemu_pci_db_modify_value.....	276
doca_devemu_pci_db_query_value.....	277
doca_devemu_pci_db_start.....	277
doca_devemu_pci_db_stop.....	278
doca_devemu_pci_dev_as_ctx.....	278
doca_devemu_pci_dev_create.....	278
doca_devemu_pci_dev_destroy.....	279
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get_bar_id.....	279
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get_bar_region_start.....	280
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get_pci_dev.....	280
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_register.....	280
doca_devemu_pci_dev_event_flr_register.....	281
doca_devemu_pci_dev_event_hotplug_state_change_register.....	282
doca_devemu_pci_dev_get_hotplug_state.....	283
doca_devemu_pci_dev_hotplug.....	283
doca_devemu_pci_dev_hotunplug.....	284
doca_devemu_pci_dev_is_flr.....	284
doca_devemu_pci_dev_modify_bar_stateful_region_default_values.....	285
doca_devemu_pci_dev_modify_bar_stateful_region_values.....	286
doca_devemu_pci_dev_query_bar_stateful_region_values.....	286
doca_devemu_pci_mmap_create.....	287
doca_devemu_pci_msix_create_on_dpa.....	288
doca_devemu_pci_msix_destroy.....	288
doca_devemu_pci_msix_get_dpa_handle.....	289
2.10.1.1. DOCA Device Emulation - PCI Device Types.....	289
2.10.2. DOCA Device Emulation - Virtio FS Devices.....	325
DOCA Device Emulation - Virtio FS IO Context.....	325
DOCA Device Emulation - Virtio FS Device Types.....	325
doca_devemu_vfs_dev_as_ctx.....	326
doca_devemu_vfs_dev_as_pci_dev.....	326
doca_devemu_vfs_dev_as_virtio_dev.....	326
doca_devemu_vfs_dev_create.....	327
doca_devemu_vfs_dev_destroy.....	327

doca_devemu_vfs_dev_get_notify_buf_size.....	328
doca_devemu_vfs_dev_get_num_request_queues.....	328
doca_devemu_vfs_dev_get_tag.....	329
doca_devemu_vfs_dev_get_vfs_notification_req_user_data_size.....	329
doca_devemu_vfs_dev_get_vfs_req_user_data_size.....	330
doca_devemu_vfs_dev_set_notify_buf_size.....	330
doca_devemu_vfs_dev_set_num_request_queues.....	331
doca_devemu_vfs_dev_set_tag.....	331
doca_devemu_vfs_dev_set_vfs_notification_req_user_data_size.....	332
doca_devemu_vfs_dev_set_vfs_req_user_data_size.....	332
DOCA_VFS_TAG_SIZE.....	333
2.10.2.1. DOCA Device Emulation - Virtio FS IO Context.....	333
2.10.2.2. DOCA Device Emulation - Virtio FS Device Types.....	339
2.10.3. DOCA Device Emulation - Virtio Devices.....	341
DOCA Device Emulation - Virtio IO Context.....	341
DOCA Device Emulation - Virtio Device Types.....	341
doca_devemu_virtio_dev_event_reset_handler_cb_t.....	342
doca_devemu_virtio_dev_as_ctx.....	342
doca_devemu_virtio_dev_as_pci_dev.....	342
doca_devemu_virtio_dev_event_reset_register.....	343
doca_devemu_virtio_dev_get_config_generation.....	343
doca_devemu_virtio_dev_get_config_msix_vector.....	344
doca_devemu_virtio_dev_get_device_features_63_0.....	344
doca_devemu_virtio_dev_get_device_status.....	345
doca_devemu_virtio_dev_get_driver_features_63_0.....	345
doca_devemu_virtio_dev_get_num_enabled_queues.....	345
doca_devemu_virtio_dev_get_num_queues.....	346
doca_devemu_virtio_dev_get_num_required_running_virtio_io_ctxs.....	346
doca_devemu_virtio_dev_get_queue_size.....	347
doca_devemu_virtio_dev_reset_complete.....	347
doca_devemu_virtio_dev_set_device_features_63_0.....	347
doca_devemu_virtio_dev_set_num_queues.....	348
doca_devemu_virtio_dev_set_num_required_running_virtio_io_ctxs.....	348
doca_devemu_virtio_dev_set_queue_size.....	349
2.10.3.1. DOCA Device Emulation - Virtio IO Context.....	349
2.10.3.2. DOCA Device Emulation - Virtio Device Types.....	349
2.11. DOCA DMA Engine.....	352
doca_dma_task_memcpy_completion_cb_t.....	352

doca_dma_as_ctx.....	353
doca_dma_cap_get_max_num_tasks.....	353
doca_dma_cap_task_memcpy_get_max_buf_list_len.....	354
doca_dma_cap_task_memcpy_get_max_buf_size.....	354
doca_dma_cap_task_memcpy_is_supported.....	355
doca_dma_create.....	355
doca_dma_destroy.....	356
doca_dma_get_gpu_handle.....	356
doca_dma_task_memcpy_alloc_init.....	357
doca_dma_task_memcpy_as_task.....	357
doca_dma_task_memcpy_get_dst.....	358
doca_dma_task_memcpy_get_src.....	358
doca_dma_task_memcpy_set_conf.....	358
doca_dma_task_memcpy_set_dst.....	359
doca_dma_task_memcpy_set_src.....	359
2.12. DOCA DPA Host.....	359
doca_dpa_dev_log_level_t.....	360
doca_dpa_dev_async_ops_t.....	360
doca_dpa_dev_completion_t.....	360
doca_dpa_dev_hash_table_t.....	360
doca_dpa_dev_notification_completion_t.....	360
doca_dpa_dev_t.....	360
doca_dpa_dev_uintptr_t.....	360
doca_dpa_func_t.....	361
doca_dpa_app_get_name.....	361
doca_dpa_async_ops_attach.....	362
doca_dpa_async_ops_create.....	362
doca_dpa_async_ops_destroy.....	363
doca_dpa_async_ops_get_dpa_handle.....	364
doca_dpa_async_ops_get_queue_size.....	364
doca_dpa_async_ops_get_user_data.....	365
doca_dpa_async_ops_start.....	365
doca_dpa_async_ops_stop.....	366
doca_dpa_cap_is_supported.....	366
doca_dpa_completion_create.....	367
doca_dpa_completion_destroy.....	367
doca_dpa_completion_get_dpa_handle.....	368
doca_dpa_completion_get_queue_size.....	368

doca_dpa_completion_get_thread.....	369
doca_dpa_completion_set_thread.....	369
doca_dpa_completion_start.....	370
doca_dpa_completion_stop.....	370
doca_dpa_create.....	371
doca_dpa_d2h_buf_memcpy.....	371
doca_dpa_d2h_memcpy.....	372
doca_dpa_destroy.....	373
doca_dpa_device_extend.....	373
doca_dpa_eu_affinity_clear.....	374
doca_dpa_eu_affinity_create.....	375
doca_dpa_eu_affinity_destroy.....	375
doca_dpa_eu_affinity_get.....	376
doca_dpa_eu_affinity_set.....	376
doca_dpa_get_app.....	377
doca_dpa_get_core_num.....	377
doca_dpa_get_dpa_handle.....	378
doca_dpa_get_kernel_max_run_time.....	378
doca_dpa_get_log_level.....	379
doca_dpa_get_max_threads_per_kernel.....	379
doca_dpa_get_num_eus_per_core.....	380
doca_dpa_get_total_num_eus_available.....	380
doca_dpa_h2d_buf_memcpy.....	381
doca_dpa_h2d_memcpy.....	381
doca_dpa_hash_table_create.....	382
doca_dpa_hash_table_destroy.....	383
doca_dpa_hash_table_get_dpa_handle.....	383
doca_dpa_kernel_launch_update_add.....	384
doca_dpa_kernel_launch_update_set.....	385
doca_dpa_log_file_get_path.....	386
doca_dpa_log_file_set_path.....	387
doca_dpa_mem_alloc.....	387
doca_dpa_mem_free.....	388
doca_dpa_memset.....	389
doca_dpa_notification_completion_create.....	389
doca_dpa_notification_completion_destroy.....	390
doca_dpa_notification_completion_get_dpa_handle.....	391
doca_dpa_notification_completion_get_thread.....	391

doca_dpa_notification_completion_start.....	392
doca_dpa_notification_completion_stop.....	392
doca_dpa_peek_at_last_error.....	392
doca_dpa_rpc.....	393
doca_dpa_set_app.....	394
doca_dpa_set_log_level.....	394
doca_dpa_start.....	395
doca_dpa_stop.....	395
doca_dpa_thread_create.....	396
doca_dpa_thread_destroy.....	396
doca_dpa_thread_get_affinity.....	397
doca_dpa_thread_get_func_arg.....	397
doca_dpa_thread_get_local_storage.....	398
doca_dpa_thread_group_create.....	398
doca_dpa_thread_group_destroy.....	399
doca_dpa_thread_group_get_num_threads.....	399
doca_dpa_thread_group_set_thread.....	400
doca_dpa_thread_group_start.....	400
doca_dpa_thread_group_stop.....	401
doca_dpa_thread_run.....	401
doca_dpa_thread_set_affinity.....	402
doca_dpa_thread_set_func_arg.....	402
doca_dpa_thread_set_local_storage.....	403
doca_dpa_thread_start.....	404
doca_dpa_thread_stop.....	404
doca_dpa_trace_file_get_path.....	405
doca_dpa_trace_file_set_path.....	405
DOCA_DPA_COMPLETION_LOG_MAX_USER_DATA.....	406

Chapter 1. Change Log

This chapter lists 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:

- ▶ [DOCA AES-GCM Engine](#)
- ▶ [DOCA App Shield](#)
 - ▶ [DOCA App Shield Attributes](#)
- ▶ [DOCA Arg Parser](#)
- ▶ [DOCA Core](#)
 - ▶ [operations for DOCA Types](#)
 - ▶ [DOCA Buffer](#)
 - ▶ [DOCA Buffer Array](#)
 - ▶ [DOCA Buffer Inventory](#)
 - ▶ [DOCA Buffer Pool](#)
 - ▶ [DOCA Context](#)
 - ▶ [DOCA Device](#)
- ▶ [DOCA Comch](#)
 - ▶ [DOCA Comch Consumer](#)
 - ▶ [DOCA Comch MsgQ](#)
 - ▶ [DOCA Comch Producer](#)
- ▶ [DOCA Comm Channel](#)
- ▶ [DOCA Compatibility Management](#)
- ▶ [DOCA Compress Engine](#)
- ▶ [DOCA Environment Configurations](#)
- ▶ [DOCA Device Emulation](#)
 - ▶ [DOCA Device Emulation - PCI Devices](#)
 - ▶ [DOCA Device Emulation - PCI Device Types](#)
 - ▶ [DOCA Device Emulation - Virtio FS Devices](#)
 - ▶ [DOCA Device Emulation - Virtio FS IO Context](#)

- ▶ [DOCA Device Emulation - Virtio FS Device Types](#)
- ▶ [DOCA Device Emulation - Virtio Devices](#)
 - ▶ [DOCA Device Emulation - Virtio IO Context](#)
 - ▶ [DOCA Device Emulation - Virtio Device Types](#)
- ▶ [DOCA DMA Engine](#)
- ▶ [DOCA DPA Host](#)

2.1. DOCA AES-GCM Engine

DOCA AES-GCM library. For more details please refer to the user guide on DOCA devzone.

enum `doca_aes_gcm_key_type`

AES-GCM key type.

Values

DOCA_AES_GCM_KEY_128 = 1

key size of 128 bit

DOCA_AES_GCM_KEY_256 = 2

key size of 256 bit

typedef

```
(*doca_aes_gcm_task_decrypt_completion_cb_t)
(doca_aes_gcm_task_decrypt* task, union doca_data
task_user_data, union doca_data ctx_user_data)
```

Function to execute on aes_gcm decrypt task completion.

typedef

```
(*doca_aes_gcm_task_encrypt_completion_cb_t)
(doca_aes_gcm_task_encrypt* task, union doca_data
task_user_data, union doca_data ctx_user_data)
```

Function to execute on aes_gcm encrypt task completion.

```
DOCA_EXPERIMENTAL doca_ctx
*doca_aes_gcm_as_ctx (doca_aes_gcm *aes_gcm)
```

Parameters

aes_gcm

AES-GCM instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Adapt doca_aes_gcm instance into a generalized context for use with doca core objects.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_get_max_num_tasks
(doca_aes_gcm *aes_gcm, uint32_t
*max_num_tasks)
```

Parameters

aes_gcm

AES-GCM context to get max number of tasks from

max_num_tasks

Sum of num tasks should not exceed this number (

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 number of tasks

This method retrieves the maximum number of tasks for a device. Sum of num tasks should not exceed this number.

See also:

doca_aes_gcm_set_aes_gcm_encrypt_task_conf,
doca_aes_gcm_set_aes_gcm_decrypt_task_conf)

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_get_max_buf_size
(const doca_devinfo *devinfo, uint64_t
*max_buffer_size)
```

Get aes_gcm decrypt max buffer size.

Parameters

devinfo

doca device info to check

max_buffer_size

The max buffer size for aes_gcm decrypt operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

This method retrieves aes_gcm decrypt max size

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_get_max_iv_len
(const doca_devinfo *devinfo, uint32_t *max_iv_len)
```

Parameters

devinfo

doca device info to check

max_iv_len

The max iv length in bytes.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support decrypt task or failed to query device capabilities.

Description

Get aes_gcm decrypt maximum initialization vector length for a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

Get the maximum supported number of elements in DOCA linked-list buffer for aes_gcm decrypt task.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_is_key_type_supported
(const doca_devinfo *devinfo,
doca_aes_gcm_key_type key_type)
```

Parameters

devinfo

doca device info to check

key_type

key type to check. See enum `doca_aes_gcm_key_type`.

Returns

DOCA_SUCCESS - in case device supports the AES-GCM key type for decrypt task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - AES-GCM key type for decrypt task is not supported by the device or devinfo does not support decrypt task or failed to query device capabilities.

Description

Check if a given AES-GCM key type for decrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_is_supported
(const doca_devinfo *devinfo)
```

Check if a aes_gcm decrypt task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

This method checks if a aes_gcm decrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_is_tag_128_supported
(const doca_devinfo *devinfo)
```

Parameters

devinfo

doxa device info to check

Returns

DOCA_SUCCESS - in case device supports authentication tag of size 128-bit
Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - authentication tag of size 128-bit is not supported by the device or devinfo does not support decrypt task or failed to query device capabilities.

Description

Check if authentication tag of size 128-bit for decrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_decrypt_is_tag_96_supported
(const doca_devinfo *devinfo)
```

Parameters

devinfo

doxa device info to check

Returns

DOCA_SUCCESS - in case device supports authentication tag of size 96-bit
Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - authentication tag of size 96-bit is not supported by the device or devinfo does not support decrypt task or failed to query device capabilities.

Description

Check if authentication tag of size 96-bit for decrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_get_max_buf_size
(const doca_devinfo *devinfo, uint64_t
*max_buffer_size)
```

Get aes_gcm encrypt max buffer size.

Parameters

devinfo

doca device info to check

max_buffer_size

The max buffer size for aes_gcm encrypt operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

This method retrieves a aes_gcm encrypt max size for a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_get_max_iv_len
(const doca_devinfo *devinfo, uint32_t *max_iv_len)
```

Parameters

devinfo

doca device info to check

max_iv_len

The max iv length in bytes.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support encrypt task or failed to query device capabilities.

Description

Get aes_gcm encrypt maximum initialization vector length for a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

Get the maximum supported number of elements in DOCA linked-list buffer for aes_gcm encrypt task.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_is_key_type_supported
(const doca_devinfo *devinfo,
doca_aes_gcm_key_type key_type)
```

Parameters

devinfo

doca device info to check

key_type

key type to check. See enum `doca_aes_gcm_key_type`.

Returns

DOCA_SUCCESS - in case device supports the AES-GCM key type for encrypt task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - AES-GCM key type for encrypt task is not supported by the device or devinfo does not support encrypt task or failed to query device capabilities.

Description

Check if a given AES-GCM key type for encrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_is_supported
(const doca_devinfo *devinfo)
```

Check if a aes_gcm encrypt task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task or failed to query device capabilities.

Description

This method checks if a aes_gcm encrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_is_tag_128_supported
(const doca_devinfo *devinfo)
```

Parameters

devinfo

doxa device info to check

Returns

DOCA_SUCCESS - in case device supports authentication tag of size 128-bit
Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - authentication tag of size 128-bit is not supported by the device or devinfo does not support encrypt task or failed to query device capabilities.

Description

Check if authentication tag of size 128-bit for encrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_cap_task_encrypt_is_tag_96_supported
(const doca_devinfo *devinfo)
```

Parameters

devinfo

doxa device info to check

Returns

DOCA_SUCCESS - in case device supports authentication tag of size 96-bit
Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - authentication tag of size 96-bit is not supported by the device or devinfo does not support encrypt task or failed to query device capabilities.

Description

Check if authentication tag of size 96-bit for encrypt task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_create (doca_dev *dev,
doca_aes_gcm **aes_gcm)
```

Parameters

dev

The device to attach to the aes_gcm context

aes_gcm

Pointer to pointer to be set to point to the created doca_aes_gcm instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - one or more of the arguments is null.
- ▶ DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_aes_gcm.

Description

Create a DOCA AES-GCM instance.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_destroy (doca_aes_gcm *aes_gcm)
```

Parameters

aes_gcm

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 aes_gcm instance or if there are undestroyed DOCA AES-GCM keys.
- ▶ DOCA_ERROR_BAD_STATE - aes_gcm context is not in idle state, try to stop the context.

Description

Destroy a DOCA AES-GCM instance.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_key_create (doca_aes_gcm
*aes_gcm, const void *raw_key,
doca_aes_gcm_key_type raw_key_type,
doca_aes_gcm_key **key)
```

Create an AES-GCM key from the user raw key to send with the task to allow encrypt/decrypt operations.

Parameters

aes_gcm

AES_GCM instance.

raw_key

The raw key given by the user, only 128bit or 256bit keys are supported

raw_key_type

The raw key type given by the user. See enum `doca_aes_gcm_key_type`.

key

Pointer to pointer to be set to point to the created AES-GCM key to allow encrypt/decrypt operations.

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 - if key type is not supported by the device or if failed to create DOCA AES-GCM key.

Description



Note:

Need to attach device to ctx before calling this function

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_key_destroy (doca_aes_gcm_key
*key)
```

Destroy AES-GCM key that was created in `doca_aes_gcm_key_create`.

Parameters

key

The AES-GCM key to allow encrypt/decrypt operations.

Returns

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

- ▶ DOCA_ERROR_DRIVER - low level layer failure.

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_task_decrypt_alloc_init
(doca_aes_gcm *aes_gcm, const doca_buf *src_buff,
doca_buf *dst_buff, doca_aes_gcm_key *key,
const uint8_t *iv, uint32_t iv_length, uint32_t
tag_size, uint32_t aad_size, doca_data user_data,
doca_aes_gcm_task_decrypt **task)
```

Allocate `aes_gcm` decrypt task.

Parameters

aes_gcm

The `aes_gcm` context to allocate the task from

src_buff

Source buffer

dst_buff

Destination buffer

key

DOCA AES-GCM key

iv

Initialization vector

iv_length

Initialization vector length in bytes, 0B-12B values are supported

tag_size

Authentication tag size in bytes, only 12B and 16B values are supported

aad_size

Additional authenticated data size in bytes

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

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 - all aes_gcm decrypt tasks are already allocated.
- ▶ DOCA_ERROR_BAD_STATE - aes_gcm context is not in running state, try to start the context.

Description

This method allocates and initializes a aes_gcm decrypt task. Task parameters can be set later on by setters.

```
DOCA_EXPERIMENTAL doca_task
*doca_aes_gcm_task_decrypt_as_task
(doca_aes_gcm_task_decrypt *task)
```

convert aes_gcm decrypt task to doca_task

Parameters**task**

The task to convert

Returns

doca_task

```
DOCA_EXPERIMENTAL uint32_t
doca_aes_gcm_task_decrypt_get_aad_size (const
doca_aes_gcm_task_decrypt *task)
```

get aes_gcm decrypt task additional authenticated data size

Parameters

task

Task to get the additional authenticated data size from

Returns

additional authenticated data size in bytes

```
DOCA_EXPERIMENTAL doca_buf
*doca_aes_gcm_task_decrypt_get_dst (const
doca_aes_gcm_task_decrypt *task)
```

get aes_gcm decrypt task destination

Parameters

task

Task to get the destination from

Returns

destination buffer

```
const DOCA_EXPERIMENTAL uint8_t
*doca_aes_gcm_task_decrypt_get_iv (const
doca_aes_gcm_task_decrypt *task, uint32_t
*iv_length)
```

get aes_gcm decrypt task initialization vector

Parameters

task

Task to get the initialization vector from

iv_length

Initialization vector length in bytes

Returns

initialization vector

```
DOCA_EXPERIMENTAL doca_aes_gcm_key
*doca_aes_gcm_task_decrypt_get_key (const
doca_aes_gcm_task_decrypt *task)
```

get aes_gcm decrypt task doca_aes_gcm_key

Parameters

task

Task to get the doca_aes_gcm_key from

Returns

DOCA AES-GCM key.

```
const DOCA_EXPERIMENTAL doca_buf
*doca_aes_gcm_task_decrypt_get_src (const
doca_aes_gcm_task_decrypt *task)
```

get aes_gcm decrypt task source

Parameters

task

Task to get the source from

Returns

source buffer

```
DOCA_EXPERIMENTAL uint32_t
doca_aes_gcm_task_decrypt_get_tag_size (const
doca_aes_gcm_task_decrypt *task)
```

get aes_gcm decrypt task authentication tag size

Parameters

task

Task to get the authentication tag size from

Returns

authentication tag size in bytes

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_decrypt_set_aad_size
(doca_aes_gcm_task_decrypt *task, uint32_t
aad_size)
```

set aes_gcm decrypt task additional authenticated data size

Parameters

task

Task to set the additional authenticated data size to

aad_size

Additional authenticated data size in bytes

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_task_decrypt_set_conf
(doca_aes_gcm *aes_gcm,
doca_aes_gcm_task_decrypt_completion_cb_t
task_completion_cb,
doca_aes_gcm_task_decrypt_completion_cb_t
task_error_cb, uint32_t num_tasks)
```

This method sets the aes_gcm decrypt task configuration.

Parameters

aes_gcm

The aes_gcm context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of aes_gcm decrypt tasks that the context can allocate

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 - aes_gcm context is not in idle state, try to stop the context.

```
DOCA_EXPERIMENTAL void  
doca_aes_gcm_task_decrypt_set_dst  
(doca_aes_gcm_task_decrypt *task, doca_buf  
*dst_buff)
```

set aes_gcm decrypt task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

```
DOCA_EXPERIMENTAL void  
doca_aes_gcm_task_decrypt_set_iv  
(doca_aes_gcm_task_decrypt *task, const uint8_t *iv,  
uint32_t iv_length)
```

set aes_gcm decrypt task initialization vector

Parameters

task

Task to set the initialization vector to

iv

Initialization vector

iv_length

Initialization vector length in bytes

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_decrypt_set_key
(doca_aes_gcm_task_decrypt *task,
doca_aes_gcm_key *key)
```

set aes_gcm decrypt task doca_aes_gcm_key

Parameters

task

Task to set the doca_aes_gcm_key to

key

DOCA AES-GCM key

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_decrypt_set_src
(doca_aes_gcm_task_decrypt *task, const doca_buf
*src_buff)
```

set aes_gcm decrypt task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_decrypt_set_tag_size
(doca_aes_gcm_task_decrypt *task, uint32_t
tag_size)
```

set aes_gcm decrypt task authentication tag size

Parameters

task

Task to set the authentication tag size to

tag_size

Authentication tag size in bytes

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_task_encrypt_alloc_init
(doca_aes_gcm *aes_gcm, const doca_buf *src_buff,
doca_buf *dst_buff, doca_aes_gcm_key *key,
const uint8_t *iv, uint32_t iv_length, uint32_t
tag_size, uint32_t aad_size, doca_data user_data,
doca_aes_gcm_task_encrypt **task)
```

Allocate aes_gcm encrypt task.

Parameters

aes_gcm

The aes_gcm context to allocate the task from

src_buff

Source buffer

dst_buff

Destination buffer

key

DOCA AES-GCM key

iv

Initialization vector

iv_length

Initialization vector length in bytes, 0B-12B values are supported

tag_size

Authentication tag size in bytes, only 12B and 16B values are supported

aad_size

Additional authenticated data size in bytes

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

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 - all aes_gcm encrypt tasks are already allocated.
- ▶ DOCA_ERROR_BAD_STATE - aes_gcm context is not in running state, try to start the context.

Description

This method allocates and initializes a aes_gcm encrypt task. Task parameters can be set later on by setters.

```
DOCA_EXPERIMENTAL doca_task
*doca_aes_gcm_task_encrypt_as_task
(doca_aes_gcm_task_encrypt *task)
```

convert aes_gcm encrypt task to doca_task

Parameters

task

The task to convert

Returns

doca_task

```
DOCA_EXPERIMENTAL uint32_t
doca_aes_gcm_task_encrypt_get_aad_size (const
doca_aes_gcm_task_encrypt *task)
```

get aes_gcm encrypt task additional authenticated data size

Parameters

task

Task to get the additional authenticated data size from

Returns

additional authenticated data size in bytes

```
DOCA_EXPERIMENTAL doca_buf
*doca_aes_gcm_task_encrypt_get_dst (const
doca_aes_gcm_task_encrypt *task)
```

get aes_gcm encrypt task destination

Parameters

task

Task to get the destination from

Returns

destination buffer

```

const DOCA_EXPERIMENTAL uint8_t
*doca_aes_gcm_task_encrypt_get_iv (const
doca_aes_gcm_task_encrypt *task, uint32_t
*iv_length)

```

get aes_gcm encrypt task initialization vector

Parameters

task

Task to get the initialization vector from

iv_length

Initialization vector length in bytes

Returns

initialization vector

```

DOCA_EXPERIMENTAL doca_aes_gcm_key
*doca_aes_gcm_task_encrypt_get_key (const
doca_aes_gcm_task_encrypt *task)

```

get aes_gcm encrypt task doca_aes_gcm_key

Parameters

task

Task to get the doca_aes_gcm_key from

Returns

DOCA AES-GCM key.

```
const DOCA_EXPERIMENTAL doca_buf
*doca_aes_gcm_task_encrypt_get_src (const
doca_aes_gcm_task_encrypt *task)
```

get aes_gcm encrypt task source

Parameters

task

Task to get the source from

Returns

source buffer

```
DOCA_EXPERIMENTAL uint32_t
doca_aes_gcm_task_encrypt_get_tag_size (const
doca_aes_gcm_task_encrypt *task)
```

get aes_gcm encrypt task authentication tag size

Parameters

task

Task to get the authentication tag size from

Returns

authentication tag size in bytes

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_aad_size
(doca_aes_gcm_task_encrypt *task, uint32_t
aad_size)
```

set aes_gcm encrypt task additional authenticated data size

Parameters

task

Task to set the additional authenticated data size to

aad_size

Additional authenticated data size in bytes

```
DOCA_EXPERIMENTAL doca_error_t
doca_aes_gcm_task_encrypt_set_conf
(doca_aes_gcm *aes_gcm,
doca_aes_gcm_task_encrypt_completion_cb_t
task_completion_cb,
doca_aes_gcm_task_encrypt_completion_cb_t
task_error_cb, uint32_t num_tasks)
```

This method sets the aes_gcm encrypt task configuration.

Parameters

aes_gcm

The aes_gcm context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of aes_gcm encrypt tasks that the context can allocate

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 - aes_gcm context is not in idle state, try to stop the context.

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_dst
(doca_aes_gcm_task_encrypt *task, doca_buf
*dst_buff)
```

set aes_gcm encrypt task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_iv
(doca_aes_gcm_task_encrypt *task, const uint8_t *iv,
uint32_t iv_length)
```

set aes_gcm encrypt task initialization vector

Parameters

task

Task to set the initialization vector to

iv

Initialization vector

iv_length

Initialization vector length in bytes

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_key
(doca_aes_gcm_task_encrypt *task,
doca_aes_gcm_key *key)
```

set aes_gcm encrypt task doca_aes_gcm_key

Parameters

task

Task to set the doca_aes_gcm_key to

key

DOCA AES-GCM key

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_src
(doca_aes_gcm_task_encrypt *task, const doca_buf
*src_buff)
```

set aes_gcm encrypt task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

```
DOCA_EXPERIMENTAL void
doca_aes_gcm_task_encrypt_set_tag_size
(doca_aes_gcm_task_encrypt *task, uint32_t
tag_size)
```

set aes_gcm encrypt task authentication tag size

Parameters**task**

Task to set the authentication tag size to

tag_size

Authentication tag size in bytes

2.2. DOCA 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. Please follow the programmer guide for system configurations.

DOCA App Shield Attributes

```
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_container_info_get
(doca_apsh_container *container,
doca_apsh_container_attr attr)
```

Shadow function - get attribute value for a container.

Parameters

container

single container handler

attr

Attribute to get the info on the container

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use `doca_apsh_container_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_injection_detect_info_get
(doca_apsh_injection_detect *suspected_injection,
doca_apsh_injection_detect_attr attr)
```

Shadow function - get attribute value for a suspected_injection.

Parameters

suspected_injection

single injection_detect handler

attr

Attribute to get the info on the suspected injection

Returns

return the info requested, need to cast

Description

Do not use this function, recommended to use `doca_apsh_injection_detect_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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_apsh_attestation_get (doca_apsh_process
*process, const char *exec_hash_map_path,
doca_apsh_attestation attestation, 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_EXPERIMENTAL doca_error_t
doca_apsh_attst_refresh
(doca_apsh_attestation, 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_error_t
doca_apsh_container_processes_get
(doca_apsh_container *container,
doca_apsh_processprocesses, int *processes_size)
```

Get array of current processes running on the container.

Parameters

container

single container 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_EXPERIMENTAL void
doca_apsh_containers_free (doca_apsh_container
**containers)
```

Destroys a container context.

Parameters

containers

Array of container opaque pointers of the systems to destroy

```
DOCA_EXPERIMENTAL doca_error_t
doca_apsh_containers_get (doca_apsh_system
*system, doca_apsh_containercontainers, int
*containers_size)
```

Get array of current containers running on the system.

Parameters

system

System handler

containers

Array of container opaque pointers of the systems

containers_size

Output param, will contain size of containers 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 containers list initialization failed.
- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to containers 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.

currently supports the following container runtime tools:

- ▶ containerd

```
DOCA_EXPERIMENTAL doca_apsh_ctx
*doca_apsh_create (void)
```

Create a new apsh handler.

Returns

apsh context required for creating system handler, NULL on failure

Description

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

DOCA_EXPERIMENTAL `void doca_apsh_destroy (doca_apsh_ctx *ctx)`

Free the APSH memory and close connections.

Parameters

ctx

apsh context to destroy

DOCA_EXPERIMENTAL `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_EXPERIMENTAL doca_error_t
doca_apsh_envars_get (doca_apsh_process *process,
doca_apsh_envarenavars, 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_EXPERIMENTAL 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_injection_detect_free
(doca_apsh_injection_detect **suspected_injections)
```

Destroys an injection_detect context.

Parameters

suspected_injections

suspected_injections opaque pointer of the process to destroy

```
DOCA_EXPERIMENTAL doca_error_t
doca_apsh_injection_detect_get
(doca_apsh_process *process,
doca_apsh_injection_detects suspected_injections, int
*suspected_injections_size)
```

Get suspected code injections of current process.

Parameters

process

Process handler

suspected_injections

suspected injections opaque pointers of the process

suspected_injections_size

Output param, will contain size of suspected_injections 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_NOT_FOUND - if process structures haven't been found.
- ▶ DOCA_ERROR_INITIALIZATION - if injections list initialization failed.
- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to injections array.
- ▶ DOCA_ERROR_NOT_SUPPORTED - in case of unsupported system os.

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.



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_EXPERIMENTAL doca_error_t
doca_apsh_ldrmodules_get (doca_apsh_process
*process, doca_apsh_ldrmodule ldrmodules, 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.



Note:

currently supported only for windows systems.

DOCA_EXPERIMENTAL void doca_apsh_libs_free (doca_apsh_lib **libs)

Destroys a libs context.

Parameters

libs

Array of libs opaque pointers of the process to destroy

DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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.
- ▶ 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).

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_EXPERIMENTAL 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 doca_error_t
doca_apsh_process_netscan_get
(doca_apsh_process *process,
doca_apsh_netscanconnections, int
*connections_size)
```

Get array of current connections for a specified process.

Parameters

process

Process 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.

- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc memory to connections array.
- ▶ DOCA_ERROR_NOT_SUPPORTED - if windows OS build is unsupported.

```

↑ list of supported windows 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

```

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 linux and windows 10 build systems (such as: windows 10 and windows server 2019).

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_EXPERIMENTAL 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_EXPERIMENTAL 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_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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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 dependent on scan_window_size, make sure to call it after "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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_apsh_threads_get (doca_apsh_process
*process, doca_apsh_thread threads, 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_EXPERIMENTAL 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_EXPERIMENTAL 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_container_info_get
(attr##_TYPE)
(uintptr_t)__doca_apsh_container_info_get(container,
attr))
```

Get attribute value for a container.

Get the requested info from container 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_injection_detect_info_get
((attr##_TYPE)
(uintptr_t) __doca_apsh_injection_detect_info_get(suspected_injection,
attr))
```

Get attribute value for a `suspected_injection`.

Get the requested info from `suspected_injection` 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(process
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

2.2.1. DOCA App Shield Attributes

DOCA App Shield

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 = 0

attestation process id

DOCA_APSH_ATTESTATION_COMM = 1

attestation process name

DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA = 2

attestation path of memory area

DOCA_APSH_ATTESTATION_PROTECTION = 3

attestation protection

DOCA_APSH_ATTESTATION_START_ADDRESS = 4

attestation start address

DOCA_APSH_ATTESTATION_END_ADDRESS = 5

attestation end address

DOCA_APSH_ATTESTATION_PAGES_NUMBER = 6

attestation process pages count in binary file

DOCA_APSH_ATTESTATION_PAGES_PRESENT = 7

attestation pages present in memory

DOCA_APSH_ATTESTATION_MATCHING_HASHES = 8

attestation pages hash match count from pages in memory

DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT = 9

attestation hash data is present

enum doca_apsh_container_attr

doca app shield process attributes

Values

DOCA_APSH_CONTAINER_ID = 0

container id

enum doca_apsh_envar_attr

doca app shield envars attributes

Values

DOCA_APSH_ENVARS_PID = 0

envars pid

DOCA_APSH_ENVARS_VARIABLE = 2

envars variable

DOCA_APSH_ENVARS_VALUE = 3

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 = 0

handle process id

DOCA_APSH_HANDLE_VALUE = 2

handle value

DOCA_APSH_HANDLE_TABLE_ENTRY = 3

handle table entry

DOCA_APSH_HANDLE_TYPE = 4

handle type

DOCA_APSH_HANDLE_ACCESS = 5

handle access

DOCA_APSH_HANDLE_NAME = 6

handle name

enum doca_apsh_injection_detect_attr

doca app shield injection detect attributes

Values

DOCA_APSH_INJECTION_DETECT_PID

suspected injection process id

DOCA_APSH_INJECTION_DETECT_VAD_START

suspected injection VAD start address

DOCA_APSH_INJECTION_DETECT_VAD_END

suspected injection VAD end address

DOCA_APSH_INJECTION_DETECT_VAD_PROTECTION

suspected injection VAD protection

DOCA_APSH_INJECTION_DETECT_VAD_TAG

suspected injection VAD pool tag

DOCA_APSH_INJECTION_DETECT_VAD_FILE_PATH

suspected injection VAD file path

DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_START

suspected injection suspected area start

DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_END

suspected injection suspected area end

enum doca_apsh_ldrmodule_attr

doca app shield LDR-Modules attributes

Values

DOCA_APSH_LDRMODULE_PID = 0

ldrmodule process pid

DOCA_APSH_LDRMODULE_BASE_ADDRESS = 2

ldrmodule base address

DOCA_APSH_LDRMODULE_LIBRARY_PATH = 3

ldrmodule loaded library path

DOCA_APSH_LDRMODULE_WINDOWS_DLL_NAME = 1000

ldrmodule dll name

DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE = 1001

ldrmodule size of image

DOCA_APSH_LDRMODULE_WINDOWS_INLOAD = 1002

ldrmodule appear in inload list

DOCA_APSH_LDRMODULE_WINDOWS_INMEM = 1003

ldrmodule appear in inmem list

DOCA_APSH_LDRMODULE_WINDOWS_ININIT = 1004

ldrmodule appear in ininit list

enum doca_apsh_lib_attr

doca app shield lib attributes

Values

DOCA_APSH_LIB_PID = 0

lib pid

DOCA_APSH_LIB_LIBRARY_PATH = 2

lib loaded library path

DOCA_APSH_LIB_LOAD_ADRESS = 3

lib load address for both Windows and Linux

DOCA_APSH_LIB_WINDOWS_DLL_NAME = 1000

lib dll name

DOCA_APSH_LIB_WINDOWS_SIZE_OF_IMAGE = 1001

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 = 0**

module offset

DOCA_APSH_MODULES_NAME = 1

module name

DOCA_APSH_MODULES_SIZE = 2

module size

enum doca_apsh_netscan_attr

doca app shield netscan attributes

Values**DOCA_APSH_NETSCAN_PID = 0**

netscan connection process id

DOCA_APSH_NETSCAN_COMM = 1

netscan connection process name

DOCA_APSH_NETSCAN_PROTOCOL = 2

netscan connection protocol

DOCA_APSH_NETSCAN_LOCAL_ADDR = 3

netscan connection local address

DOCA_APSH_NETSCAN_REMOTE_ADDR = 4

netscan connection remote address

DOCA_APSH_NETSCAN_LOCAL_PORT = 5

netscan connection local port

DOCA_APSH_NETSCAN_REMOTE_PORT = 6

netscan connection remote port

DOCA_APSH_NETSCAN_STATE = 7

netscan connection state

DOCA_APSH_NETSCAN_TIME = 8

netscan connection creation time - windows only. deprecated - use

DOCA_APSH_NETSCAN_WINDOWS_TIME instead

DOCA_APSH_NETSCAN_WINDOWS_TIME = 1000

netscan windows connection creation time

DOCA_APSH_NETSCAN_LINUX_FD = 2000

netscan linux connection file descriptor

DOCA_APSH_NETSCAN_LINUX_SOCKET_OFFSET = 2001

netscan linux connection socket offset

DOCA_APSH_NETSCAN_LINUX_FAMILY = 2002

netscan linux connection Family

DOCA_APSH_NETSCAN_LINUX_TYPE = 2003

netscan linux connection Type

DOCA_APSH_NETSCAN_LINUX_FILTER = 2004

netscan linux connection filter

DOCA_APSH_NETSCAN_LINUX_NET_NAMESPACE = 2005

netscan linux connection net namespace

DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_SENT = 2006

netscan linux connection TCP sent bytes

DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_ACKED = 2007

netscan linux connection TCP acknowledged bytes

DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_RECEIVED = 2008

netscan linux connection TCP received bytes

DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_IN = 2009

netscan linux connection TCP segments in

DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_OUT = 2010

netscan linux connection TCP segments out

DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_IN = 2011

netscan linux connection TCP data segments in

DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_OUT = 2012

netscan linux connection TCP data segments out

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 = 0

privilege process pid

DOCA_APSH_PRIVILEGES_NAME = 2

privilege name, for example: SeTcbPrivilege

DOCA_APSH_PRIVILEGES_IS_ON = 3

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 = 1001

privilege enabled flag

DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT = 1002

privilege enabledbydefault flag

enum doca_apsh_process_attr

doca app shield process attributes

Values

DOCA_APSH_PROCESS_PID = 0

process id

DOCA_APSH_PROCESS_PPID = 1

process parent id

DOCA_APSH_PROCESS_COMM = 2

process executable name

DOCA_APSH_PROCESS_CPU_TIME = 3

process cpu time [ps]

DOCA_APSH_PROCESS_WINDOWS_OFFSET = 1000

process offset

DOCA_APSH_PROCESS_WINDOWS_THREADS = 1001

process thread count

DOCA_APSH_PROCESS_WINDOWS_EXIT_TIME = 1002

process exit time

DOCA_APSH_PROCESS_LINUX_GID = 2000

process group id

DOCA_APSH_PROCESS_LINUX_UID = 2001

process user id

DOCA_APSH_PROCESS_LINUX_STATE = 2002

process state

DOCA_APSH_PROCESS_LINUX_NS_PID = 2003

process PID namespace

DOCA_APSH_PROCESS_LINUX_NS_MNT = 2004

process mount namespace

DOCA_APSH_PROCESS_LINUX_NS_NET = 2005

process network namespace

enum doca_apsh_process_parameters_attr

doca app shield process-parameters attributes

Values

DOCA_APSH_PROCESS_PARAMETERS_PID = 0

process-parameters pid

DOCA_APSH_PROCESS_PARAMETERS_CMD_LINE = 1

process-parameters command line

DOCA_APSH_PROCESS_PARAMETERS_IMAGE_BASE_ADDR = 2

process-parameters image base address

DOCA_APSH_PROCESS_PARAMETERS_IMAGE_FULL_PATH = 3

process-parameters image full path

enum `doca_apsh_sid_attr`

doca app shield SID (security identifiers) attributes

Values

DOCA_APSH_PROCESS_SID_PID = 0

SID process id

DOCA_APSH_PROCESS_SID_STRING = 1

SID string

DOCA_APSH_PROCESS_SID_ATTRIBUTES = 2

SID attributes flag

enum `doca_apsh_system_config_attr`

doca app shield configuration attributes

Values

DOCA_APSH_OS_SYMBOL_MAP = 0

os symbol map path

DOCA_APSH_MEM_REGION = 1

memory region path

DOCA_APSH_KPGD_FILE = 2

kpgd file path

DOCA_APSH_VHCA_ID = 3

vhca id

DOCA_APSH_OS_TYPE = 4

os type

DOCA_APSH_SCAN_WIN_SIZE = 5

yara scan window size

DOCA_APSH_SCAN_WIN_STEP = 6

yara scan window step

DOCA_APSH_HASHTEST_LIMIT = 7

limit of vm areas to attest

DOCA_APSH_MODULES_LIMIT = 8

limit of modules number

DOCA_APSH_PROCESS_LIMIT = 9

limit of processes number

DOCA_APSH_THREADS_LIMIT = 10

limit of threads number

DOCA_APSH_LDRMODULES_LIMIT = 11

limit of ldrmodules number on windows

DOCA_APSH_LIBS_LIMIT = 12

limit of libs number

DOCA_APSH_VADS_LIMIT = 13

limit of vads number

DOCA_APSH_WINDOWS_ENVARS_LIMIT = 14

length limit of envars for windows

DOCA_APSH_HANDLES_LIMIT = 15

limit of handles number on windows

DOCA_APSH_STRING_LIMIT = 16

length limit of apsh_read_str

enum doca_apsh_system_os

system os types

Values

DOCA_APSH_SYSTEM_LINUX = 0

linux

DOCA_APSH_SYSTEM_WINDOWS = 1

windows

enum doca_apsh_thread_attr

doca app shield thread attributes

Values

DOCA_APSH_THREAD_PID = 0

thread process id

DOCA_APSH_THREAD_TID = 1

thread id

DOCA_APSH_THREAD_STATE = 2

thread state

DOCA_APSH_THREAD_WINDOWS_WAIT_REASON = 1000

thread wait reason

DOCA_APSH_THREAD_WINDOWS_OFFSET = 1001

thread offset

DOCA_APSH_THREAD_WINDOWS_SUSPEND_COUNT = 1002

thread suspend count

DOCA_APSH_THREAD_LINUX_PROC_NAME = 2000

thread process name

DOCA_APSH_THREAD_LINUX_THREAD_NAME = 2001

thread name

enum doca_apsh_vad_attr

doca app shield virtual address descriptor attributes

Values

DOCA_APSH_VMA_PID = 0

vma process id

DOCA_APSH_VMA_OFFSET = 1

vma offset

DOCA_APSH_VMA_PROTECTION = 2

vma protection

DOCA_APSH_VMA_VM_START = 3

vma vm start

DOCA_APSH_VMA_VM_END = 4

vma vm end

DOCA_APSH_VMA_PROCESS_NAME = 5

vma process name

DOCA_APSH_VMA_FILE_PATH = 6

vma file path

DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE = 1000

vma commit charge

DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY = 1001

vma private memory

DOCA_APSH_VMA_WINDOWS_TAG = 1002

vma pool tag

enum doca_apsh_yara_attr

doca app shield yara attributes

Values

DOCA_APSH_YARA_PID = 0

pid of the process

DOCA_APSH_YARA_COMM = 1

name of the process

DOCA_APSH_YARA_RULE = 2

rule name

DOCA_APSH_YARA_MATCH_WINDOW_ADDR = 3

virtual address of the scan window of the match

DOCA_APSH_YARA_MATCH_WINDOW_LEN = 4

length of the scan window of the match

enum doca_apsh_yara_rule

available doca app shield yara rules

Values

DOCA_APSH_YARA_RULE_HELLO_WORLD = 0

yara rule that scans for "Hello World". Rule name is "Hello_World".

DOCA_APSH_YARA_RULE_REFLECTIVE_DLL_INJECTION = 1

yara rule that scans for Reflective DLL Injection attack. Rule name is "Reflective_DLL_Injection".

DOCA_APSH_YARA_RULE_MIMIKATZ = 2

yara rule that scans for Mimikatz 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 uint32_t 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 char *DOCA_APSH_CONTAINER_ID_TYPE
```

container id type

```
typedef doca_dev *DOCA_APSH_DMA_DEV_TYPE
```

dma dev name

```
typedef uint32_t 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_NAME_TYPE
```

handle name type

```
typedef uint32_t 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 uint32_t  
DOCA_APSH_INJECTION_DETECT_PID_TYPE
```

injection detect pid type

```
typedef uint64_t  
DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_END_TYPE
```

injection detect suspected area end type

```
typedef uint64_t  
DOCA_APSH_INJECTION_DETECT_SUSPECTED_AREA_START_TYPE
```

injection detect suspected area start type

```
typedef uint64_t  
DOCA_APSH_INJECTION_DETECT_VAD_END_TYPE
```

injection detect VAD end address type

typedef char

*DOCA_APSH_INJECTION_DETECT_VAD_FILE_PATH_TYPE

injection detect VAD file path type

typedef char

*DOCA_APSH_INJECTION_DETECT_VAD_PROTECTION_TYPE

injection detect VAD protection type

typedef uint64_t

DOCA_APSH_INJECTION_DETECT_VAD_START_TYPE

injection detect VAD start address type

typedef char

*DOCA_APSH_INJECTION_DETECT_VAD_TAG_TYPE

injection detect VAD pool tag type

typedef char *DOCA_APSH_KPGD_FILE_TYPE

kpgd file path

typedef uint64_t

DOCA_APSH_LDRMODULE_BASE_ADDRESS_TYPE

ldrmodule base address type

typedef char

*DOCA_APSH_LDRMODULE_LIBRARY_PATH_TYPE

ldrmodule library path type

typedef uint32_t DOCA_APSH_LDRMODULE_PID_TYPE

ldrmodule pid type

typedef char

*DOCA_APSH_LDRMODULE_WINDOWS_DLL_NAME_TYPE

ldrmodule windows 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 uint32_t  
DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE_TYPE  
ldrmodule size of image type
```

```
typedef char *DOCA_APSH_LIB_LIBRARY_PATH_TYPE  
lib loaded library path type
```

```
typedef uint64_t  
DOCA_APSH_LIB_LINUX_LOAD_ADDRESS_TYPE  
lib load address for Linux
```

```
typedef uint64_t DOCA_APSH_LIB_LOAD_ADDRESS_TYPE  
lib load address for both Windows and Linux
```

```
typedef uint32_t DOCA_APSH_LIB_PID_TYPE  
lib pid type
```

```
typedef char  
*DOCA_APSH_LIB_WINDOWS_DLL_NAME_TYPE  
lib dll name type
```

```
typedef uint32_t  
DOCA_APSH_LIB_WINDOWS_SIZE_OF_IMAGE_TYPE  
lib size of image 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
```

limit 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_LINUX_FAMILY_TYPE
```

netscan linux connection Family

```
typedef uint32_t DOCA_APSH_NETSCAN_LINUX_FD_TYPE
```

netscan linux connection file descriptor

```
typedef char *DOCA_APSH_NETSCAN_LINUX_FILTER_TYPE
```

netscan linux connection filter

```
typedef uint32_t
```

```
DOCA_APSH_NETSCAN_LINUX_NET_NAMESPACE_TYPE
```

netscan linux connection net namespace

```
typedef uint64_t
```

```
DOCA_APSH_NETSCAN_LINUX_SOCKET_OFFSET_TYPE
```

netscan linux connection socket offset

```
typedef uint64_t
```

```
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_ACKED_TYPE
```

netscan linux connection TCP acknowledged bytes

```
typedef uint64_t  
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_RECEIVED_TYPE
```

netscan linux connection TCP received bytes

```
typedef uint64_t  
DOCA_APSH_NETSCAN_LINUX_TCP_BYTES_SENT_TYPE
```

netscan linux connection TCP sent bytes

```
typedef uint32_t  
DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_IN_TYPE
```

netscan linux connection TCP data segments in

```
typedef uint32_t  
DOCA_APSH_NETSCAN_LINUX_TCP_DATA_SEGS_OUT_TYPE
```

netscan linux connection TCP data segments out

```
typedef uint32_t  
DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_IN_TYPE
```

netscan linux connection TCP segments in

```
typedef uint32_t  
DOCA_APSH_NETSCAN_LINUX_TCP_SEGS_OUT_TYPE
```

netscan linux connection TCP segments out

```
typedef char *DOCA_APSH_NETSCAN_LINUX_TYPE_TYPE
```

netscan linux connection Type

```
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 protocol

```
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 windows connection creation time - deprecated

```
typedef char
```

```
*DOCA_APSH_NETSCAN_WINDOWS_TIME_TYPE
```

netscan windows connection creation time

```
typedef char *DOCA_APSH_OS_SYMBOL_MAP_TYPE
```

os symbol map path

```
typedef DOCA_APSH_OS_TYPE_TYPE
```

os type

```
typedef bool DOCA_APSH_PRIVILEGES_IS_ON_TYPE
```

privilege is on type

```
typedef char *DOCA_APSH_PRIVILEGES_NAME_TYPE
```

privilege name type

```
typedef uint32_t 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 uint32_t DOCA_APSH_PROCESS_LINUX_GID_TYPE  
process gid type
```

```
typedef uint32_t  
DOCA_APSH_PROCESS_LINUX_NS_MNT_TYPE  
process mount namespace type
```

```
typedef uint32_t  
DOCA_APSH_PROCESS_LINUX_NS_NET_TYPE  
process network namespace type
```

```
typedef uint32_t  
DOCA_APSH_PROCESS_LINUX_NS_PID_TYPE  
process PID namespace type
```

```
typedef uint64_t  
DOCA_APSH_PROCESS_LINUX_STATE_TYPE  
process state type
```

```
typedef uint32_t 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_TYPE

process-parameters image base address

typedef char

***DOCA_APSH_PROCESS_PARAMETERS_IMAGE_FULL_PATH_TYPE**

process-parameters image full path

typedef uint32_t

DOCA_APSH_PROCESS_PARAMETERS_PID_TYPE

process-parameters pid

typedef uint32_t DOCA_APSH_PROCESS_PID_TYPE

process pid type

typedef uint32_t DOCA_APSH_PROCESS_PPID_TYPE

process pid type

typedef uint32_t

DOCA_APSH_PROCESS_SID_ATTRIBUTES_TYPE

SID attributes flag.

typedef uint32_t 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_EXIT_TIME_TYPE

process exit time type

```
typedef uint64_t  
DOCA_APSH_PROCESS_WINDOWS_OFFSET_TYPE  
process offset type
```

```
typedef uint32_t  
DOCA_APSH_PROCESS_WINDOWS_THREADS_TYPE  
process threads type
```

```
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 uint32_t DOCA_APSH_THREAD_PID_TYPE  
thread pid type
```

```
typedef uint64_t DOCA_APSH_THREAD_STATE_TYPE  
thread state type
```

```
typedef uint32_t DOCA_APSH_THREAD_TID_TYPE  
thread tid type
```

```
typedef uint64_t  
DOCA_APSH_THREAD_WINDOWS_OFFSET_TYPE  
thread offset type
```

```
typedef uint8_t  
DOCA_APSH_THREAD_WINDOWS_SUSPEND_COUNT_TYPE  
thread suspend count type
```

```
typedef uint8_t  
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 uint32_t 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 uint32_t  
DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE_TYPE  
vma commit charge type
```

```
typedef uint32_t  
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
```

2.3. DOCA Arg Parser

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

```
enum doca_argp_type  
Flag input type.
```

Values

DOCA_ARGP_TYPE_UNKNOWN = 0

DOCA_ARGP_TYPE_STRING

Input type is a string

DOCA_ARGP_TYPE_INT

Input type is an integer

DOCA_ARGP_TYPE_BOOLEAN

Input type is a boolean

DOCA_ARGP_TYPE_JSON_OBJ

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

```
typedef (*doca_argp_dpdk_cb_t) (int argc, char*
*argv)
```

DPDK flags callback function type.

```
typedef (*doca_argp_param_cb_t) (void* , void* )
```

Flag callback function type.

```
typedef (*doca_argp_validation_cb_t) (void* )
```

Program validation callback function type.

```
DOCA_EXPERIMENTAL doca_error_t
```

```
doca_argp_destroy (void)
```

ARG Parser destroy.

Returns

DOCA_SUCCESS - in case of success, the relevant error otherwise.

Description

cleanup all resources including the parsed DPDK flags, the program can't use them any more.

DOCA_EXPERIMENTAL doca_error_t doca_argp_get_log_level (int *log_level)

Get the log level the user inserted it.

Parameters

log_level

The log level if the user inserted it, otherwise the default value of log level.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - invalid input received.

DOCA_EXPERIMENTAL doca_error_t doca_argp_get_sdk_log_level (int *log_level)

Get the SDK log level the user inserted it.

Parameters

log_level

The log level if the user inserted it, otherwise the default value of log level.

Description

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

- ▶ DOCA_ERROR_INVALID_VALUE - invalid input received.

DOCA_EXPERIMENTAL doca_error_t doca_argp_init (const char *program_name, void *program_config)

Initialize the parser interface.

Parameters

program_name

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

program_config

Program configuration struct.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description



Note:

After a successful call to this function, [doca_argp_destroy\(\)](#) should be called as part of program cleanup.

```
DOCA_EXPERIMENTAL doca_error_t
doca_argp_param_create (doca_argp_param
**param)
```

Create new program param.

Parameters

param

Create program param 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 enough space.

Description



Note:

Need to set param fields by setter functions.

```
DOCA_EXPERIMENTAL doca_error_t
doca_argp_param_destroy (doca_argp_param
*param)
```

Destroy a program param.

Parameters

param

The program param to destroy.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - invalid input received.

DOCA_EXPERIMENTAL void
doca_argp_param_set_arguments (doca_argp_param
***param, const char *arguments)**

Set the expected arguments of the program param, used to print the program usage.

Parameters

param

The program param.

arguments

The param's arguments.

Description



Note:

Passing a "param" value of NULL will result in an undefined behavior.

DOCA_EXPERIMENTAL void
doca_argp_param_set_callback (doca_argp_param
***param, doca_argp_param_cb_t callback)**

Set the callback function of the program param.

Parameters

param

The program param.

callback

The param's callback function.

Description



Note:

- ▶ Passing a "param" value of NULL will result in an undefined behavior.

- ▶ Once ARGP identifies this param in CLI, will call the callback function with attaching the param argument value as first argument and next the program configuration struct. Program should copy the argument value and shouldn't use it directly.
- ▶ Set param callback is mandatory.

DOCA_EXPERIMENTAL void
doca_argp_param_set_cli_only (doca_argp_param *param)

Set if the program param is supported only CLI mode and will not be used in JSON file, by default the value is false.

Parameters

param

The program param.

Description



Note:

Passing a "param" value of NULL will result in an undefined behavior.

DOCA_EXPERIMENTAL void
doca_argp_param_set_description
(doca_argp_param *param, const char *description)

Set the description of the program param, used to print the program usage.

Parameters

param

The program param.

description

The param's description.

Description



Note:

- ▶ Passing a "param" value of NULL will result in an undefined behavior.
- ▶ Set param description is mandatory.

```
DOCA_EXPERIMENTAL void  
doca_argp_param_set_long_name  
(doca_argp_param *param, const char *name)
```

Set the long name of the program param.

Parameters

param

The program param.

name

The param's long name.

Description

**Note:**

- ▶ Passing a "param" value of NULL will result in an undefined behavior.
- ▶ At least one of param names should be set.

```
DOCA_EXPERIMENTAL void  
doca_argp_param_set_mandatory (doca_argp_param  
*param)
```

Set if the program param is mandatory, by default the value is false.

Parameters

param

The program param.

Description

**Note:**

Passing a "param" value of NULL will result in an undefined behavior.

DOCA_EXPERIMENTAL void doca_argp_param_set_multiplicity (doca_argp_param *param)

Set if the program param will appear multiple times, by default the value is false.

Parameters

param

The program param.

Description



Note:

- ▶ Passing a "param" value of NULL will result in an undefined behavior.
- ▶ Since JSON file doesn't support keys multiplicity, the multi values will be in array and param argument type will indicate the values type.

DOCA_EXPERIMENTAL void doca_argp_param_set_short_name (doca_argp_param *param, const char *name)

Set the short name of the program param.

Parameters

param

The program param.

name

The param's short name

Description



Note:

- ▶ Passing a "param" value of NULL will result in an undefined behavior.
- ▶ At least one of param names should be set.

DOCA_EXPERIMENTAL void
doca_argp_param_set_type (doca_argp_param
 *param, doca_argp_type type)

Set the type of the param arguments.

Parameters

param

The program param.

type

The param arguments type.

Description



Note:

- ▶ Passing a "param" value of NULL will result in an undefined behavior.
- ▶ Set param arguments type is mandatory.

DOCA_EXPERIMENTAL doca_error_t
doca_argp_register_param (doca_argp_param
 *input_param)

Register a program flag.

Parameters

input_param

Program flag details.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_INITIALIZATION - received param with missing mandatory fields initialization.

Description



Note:

- ▶ Value of `is_cli_only` field may be changed in this function.
- ▶ ARGV takes ownership of the pointer and will free the param including in case of failure.

DOCA_EXPERIMENTAL doca_error_t doca_argp_register_validation_callback (doca_argp_validation_cb_t callback)

Register program validation callback function.

Parameters

callback

Program validation callback.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description



Note:

When calling validation callback, will pass one argument which is the program configuration struct.

DOCA_EXPERIMENTAL doca_error_t doca_argp_register_version_callback (doca_argp_param_cb_t callback)

Register an alternative version callback.

Parameters

callback

Program-specific version callback.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description



Note:

: When calling version callback the program will exit.

DOCA_EXPERIMENTAL void
doca_argp_set_dpdk_program (doca_argp_dpdk_cb_t
callback)

Set information about program configuration, if it's based on DPDK API.

Parameters

callback

Once ARGV finished parsing DPDK flags will be forwarded to the program by calling this callback.

Description



Note:

- ▶ Need to call `doca_argp_init` before setting program DPDK type.
- ▶ If program is based on DPDK API, DPDK flags array will be sent using the callback, the array will be released when calling `doca_argp_destroy`.

DOCA_EXPERIMENTAL doca_error_t doca_argp_start
(int argc, char **argv)

Parse incoming arguments (cmd line/json).

Parameters

argc

Number of program command line arguments.

argv

Program command line arguments.

Returns

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

- ▶ DOCA_ERROR_NOT_SUPPORTED - received unsupported program flag.

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_IO_FAILED - Internal errors about JSON API, reading JSON content.
- ▶ DOCA_ERROR_NO_MEMORY - failed to allocate enough space.
- ▶ DOCA_ERROR_INITIALIZATION - initialization error.

Description

**Note:**

: if the program is based on DPDK API, DPDK flags will be forwarded to it by calling the registered callback.

DOCA_EXPERIMENTAL void doca_argp_usage (void)

Print usage instructions.

Description

**Note:**

: When calling this function the program will exit.

2.4. DOCA Core

operations for DOCA Types

DOCA Buffer

DOCA Buffer Array

DOCA Buffer Inventory

DOCA Buffer Pool

DOCA Context

DOCA Device

2.4.1. operations for DOCA Types

DOCA Core

DOCA bitfield introduces bitfield operations on DOCA type that are common for many libraries.

`__doca_builtin_ffsll (long long x)`

Parameters

x

value

Returns

: the least significant 1-bit index plus one; if x is zero, return zero.

Description

`__doca_builtin_ffsll()` - internal implementation on windows, equal to gnu's `__builtin_ffsll()`;

```
#define DOCA_BE16_GENMASK
(DOCA_HTOBE16((UINT16_MAX - (UINT16_C(1) << (_l)) + 1)
& (UINT16_MAX >> (16 - 1 - (_h)))))
```

`DOCA_BE16_GENMASK()` - generate continuous mask from `_l` bit to `_h` bit, return in big endian For example, `DOCA_BE16_GENMASK(11, 4)` -> `htons(0x0FF0)`

```
#define DOCA_BE16_GET ((DOCA_BETOH16((_m) & (_f)) >>
DOCA_SHIFT(DOCA_BETOH16(_m))))
```

`DOCA_BE16_GET()` - get a bitfield segment value

`DOCA_BE16_GET()` get the field value in host endian specified by `_m` from the register passed in as `_f` by masking and shifting it down

```
#define DOCA_BE16_SET ((DOCA_HTOBE16(_v <<
DOCA_SHIFT(DOCA_BETOH16(_m)))) & _m)
```

`DOCA_BE16_SET()` - set a bitfield segment in big endian and return

DOCA_BE16_SET() mask and shift up the value and return in `doca_be16_t` The return value should be logical OR with other fields in register.

```
#define DOCA_BE16P_GENMASK do { \ *(doca_be16_t *)_p  
= DOCA_BE16_GENMASK(_h, _l); \ } while (0)
```

DOCA_BE16P_GENMASK() - generate continuous mask from `_l` bit to `_h` bit, put in `_p` pointed memory in big endian

```
#define DOCA_BE16P_SET do { \ doca_be16_t _tmp =  
*(doca_be16_t *)_p; \ \ _tmp |= DOCA_BE16_SET(_m, _v); \  
*_p = _tmp; \ } while (0);
```

DOCA_BE16P_SET() - set a bitfield segment in `_p` pointed `doca_be16_t` field

DOCA_BE16P_SET() mask and shift up the value and logical OR with other fields in `doca_be16_t`

```
#define DOCA_BE32_GENMASK  
(DOCA_HTOBE32((UINT32_MAX - (UINT32_C(1) << (_l)) + 1)  
& (UINT32_MAX >> (32 - 1 - (_h)))))
```

DOCA_BE32_GENMASK() - generate continuous mask from `_l` bit to `_h` bit, return in big endian For example, DOCA_BE32_GENMASK(23, 4) -> `htonl(0x00FFFFFF0)`

```
#define DOCA_BE32_GET ((DOCA_BETOH32((_m) & (_f)) >>  
DOCA_SHIFT(DOCA_BETOH32(_m))))
```

DOCA_BE32_GET() - get a bitfield segment value

DOCA_BE32_GET() get the field value in host endian specified by `_m` from the register passed in as `_f` by masking and shifting it down

```
#define DOCA_BE32_SET ((DOCA_HTOBE32(_v <<  
DOCA_SHIFT(DOCA_BETOH32(_m)))) & _m)
```

DOCA_BE32_SET() - set a bitfield segment in big endian and return

DOCA_BE32_SET() mask and shift up the value and return in `doca_be32_t` The return value should be logical OR with other fields in register.

```
#define DOCA_BE32P_GENMASK do { \ *(doca_be32_t *)_p  
= DOCA_BE32_GENMASK(_h, _l); \ } while (0)
```

DOCA_BE32P_GENMASK() - generate continuous mask from _l bit to _h bit, put in _p pointed memory in big endian

```
#define DOCA_BE32P_SET do { \ doca_be32_t _tmp =  
*(doca_be32_t *)_p; \ \ _tmp |= DOCA_BE32_SET(_m, _v); \  
*_p = _tmp; \ } while (0);
```

DOCA_BE32P_SET() - set a bitfield segment in _p pointed doca_be32_t field

DOCA_BE32P_SET() mask and shift up the value and logical OR with other fields in doca_be32_t

```
#define DOCA_BETOH16 _byteswap_ushort(_x)
```

DOCA_BETOH16() - convert 16bit to host endian from big endian

```
#define DOCA_BETOH32 _byteswap_ulong(_x)
```

DOCA_BETOH32() - convert 32bit to host endian from big endian

```
#define DOCA_HTOBE16 _byteswap_ushort(_x)
```

DOCA_HTOBE16() - convert 16bit type to big endian from host endian

```
#define DOCA_HTOBE32 _byteswap_ulong(_x)
```

DOCA_HTOBE32() - convert 32bit type to big endian from host endian

```
#define DOCA_SHIFT (__doca_builtin_ffsl(_x) - 1)
```

DOCA_SHIFT() - get number of bits shifted

```
#define DOCA_U8_GENMASK ((UINT8_MAX - (UINT8_C(1) <<  
[_l]) + 1) & (UINT8_MAX >> (8 - 1 - [_h])))
```

DOCA_U8_GENMASK() - generate continuous mask from _l bit to _h bit, return in host endian
For example, DOCA_U8_GENMASK(7, 4) -> 0xF0

```
#define DOCA_U8_GET (((_m) & (_f)) >> DOCA_SHIFT((_m)))
```

DOCA_U8_GET() - get a bitfield segment value

DOCA_U8_GET() get the field value in host endian specified by `_m` from the register passed in as `_f` by masking and shifting it down

```
#define DOCA_U8_SET ((_v << DOCA_SHIFT(_m)) & _m)
```

DOCA_U8_SET() - set a bitfield segment in host endian and return

DOCA_U8_SET() mask and shift up the value and return in `uint8_t` The return value should be logical OR with other fields in register.

```
#define DOCA_U8P_GENMASK do { \ *(uint8_t *)_p = \ DOCA_U8_GENMASK(_h, _l); \ } while (0)
```

DOCA_U8P_GENMASK() - generate continuous mask from `_l` bit to `_h` bit, put in `_p` pointed memory in host endian

```
#define DOCA_U8P_SET do { \ uint8_t _tmp = *(uint8_t *)_p; \ _tmp |= DOCA_U8_SET(_m, _v); \ *_p = _tmp; \ } while (0);
```

DOCA_U8P_SET() - set a bitfield segment in `_p` pointed `uint8_t` field

DOCA_U8P_SET() mask and shift up the value and logical OR with other fields in `uint8_t`

2.4.2. DOCA Buffer

DOCA 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_STABLE doca_error_t doca_buf_chain_list (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

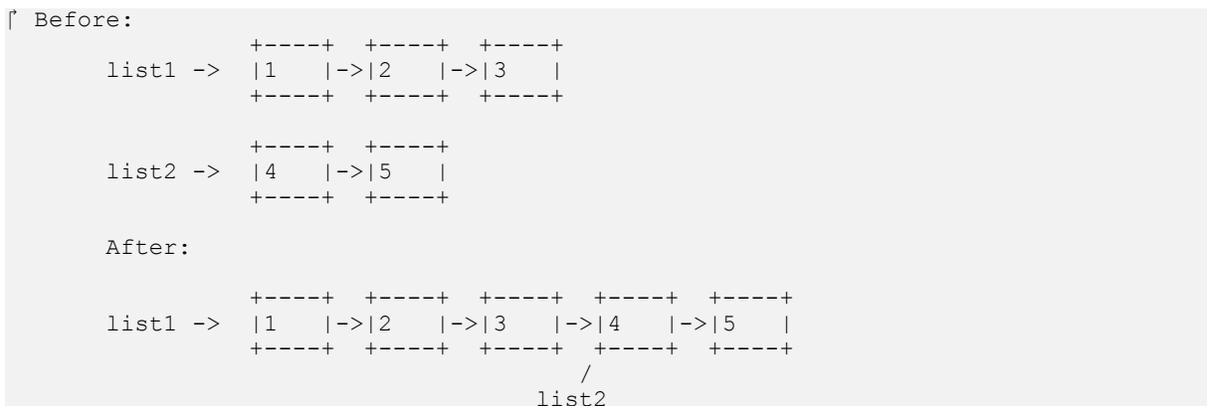
DOCA Buf representing list2. MUST NOT BE NULL AND MUST BE HEAD OF LIST. must have a refcount of 1

Returns

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

- ▶ DOCA_ERROR_NOT_PERMITTED - if list2 has a reference count that is not 1

Description



DOCA_STABLE doca_error_t doca_buf_dec_refcount (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.
- ▶ DOCA_ERROR_BAD_STATE - reference count is already 0.

Description

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



Note:

In case of list if head refcount reaches 0, then all buffers in the list will be released.

```
DOCA_STABLE doca_error_t doca_buf_get_data (const  
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_STABLE 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_STABLE 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_STABLE doca_error_t doca_buf_get_last_in_list
(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_STABLE 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_STABLE doca_error_t doca_buf_get_list_len (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_STABLE doca_error_t doca_buf_get_next_in_list
(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_STABLE doca_error_t doca_buf_get_refcount (const
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_STABLE doca_error_t doca_buf_inc_refcount
(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_SUCCESS - in case of success. doca_error code - in case of failure:

- ▶ DOCA_ERROR_NOT_PERMITTED - buf is the next element in some list.
- ▶ DOCA_ERROR_TOO_BIG - reference count already reached maximum value of UINT16_MAX.

Description



Note:

In case of list all intermediate buffers will always have a refcount of 1. As such the reference count is managed for the head only.

DOCA_STABLE doca_error_t doca_buf_is_first_in_list (const doca_buf *buf, uint8_t *is_first)

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

Parameters

buf

DOCA Buf element.

is_first

1 if buf is the first element, 0 otherwise.

Returns

DOCA_SUCCESS - always.

DOCA_STABLE doca_error_t doca_buf_is_in_list (const doca_buf *buf, uint8_t *is_in_list)

Check if provided DOCA Buf is a linked list.

Parameters

buf

DOCA Buf element. MUST NOT BE NULL.

is_in_list

1 if buf is part of a linked list, 0 if it is not. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

DOCA_STABLE doca_error_t doca_buf_is_last_in_list (const doca_buf *buf, uint8_t *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

1 if buf is the last element, 0 otherwise. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always.

DOCA_STABLE doca_error_t doca_buf_reset_data_len (doca_buf *buf)

Parameters

buf

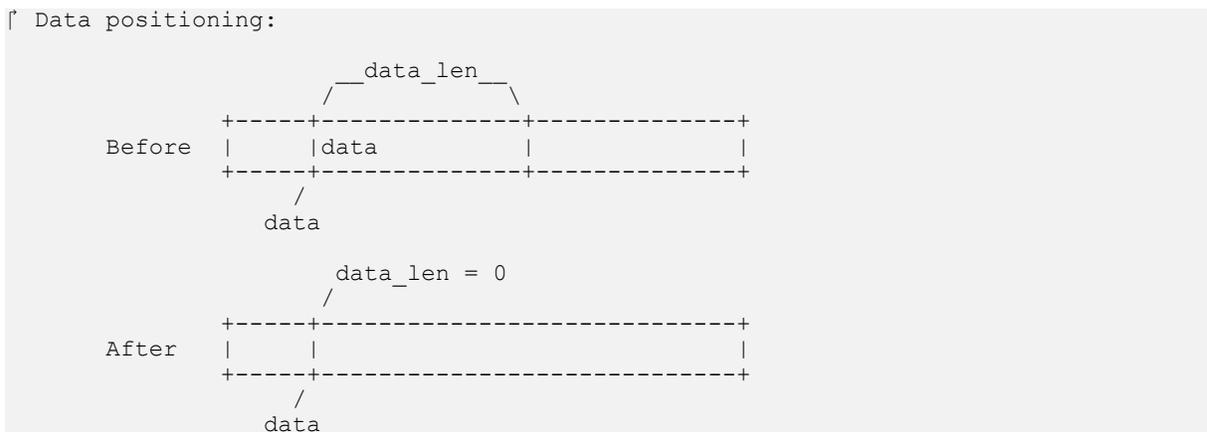
DOCA Buf element. MUST NOT BE NULL.

Returns

DOCA_SUCCESS - always

Description

Reset the data length to 0 (data will still point to the same location)



DOCA_STABLE `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.

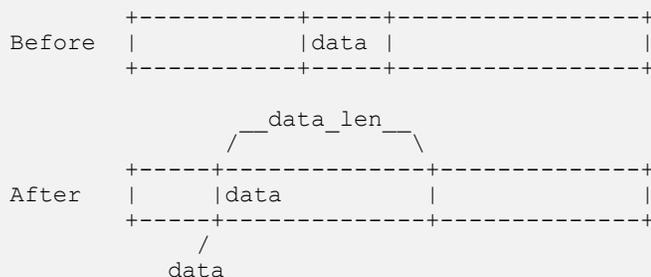
Returns

DOCA_SUCCESS - always

Description

Set data pointer and data length

↑ Data positioning:



Note:

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

DOCA_STABLE `doca_error_t doca_buf_set_data_len (doca_buf *buf, size_t data_len)`

Parameters

buf

DOCA Buf element. MUST NOT BE NULL.

data_len

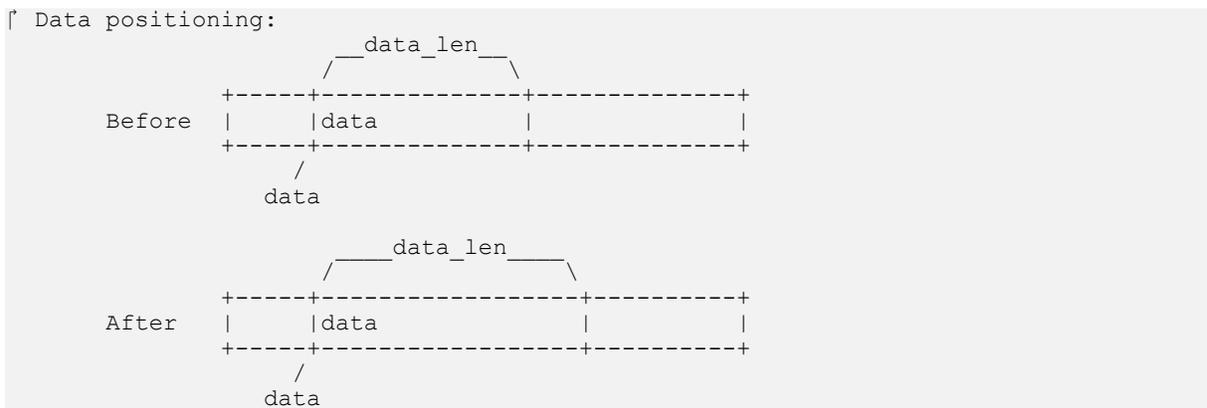
Data length.

Returns

DOCA_SUCCESS - always

Description

Set data length



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

DOCA_STABLE doca_error_t doca_buf_unchain_list (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

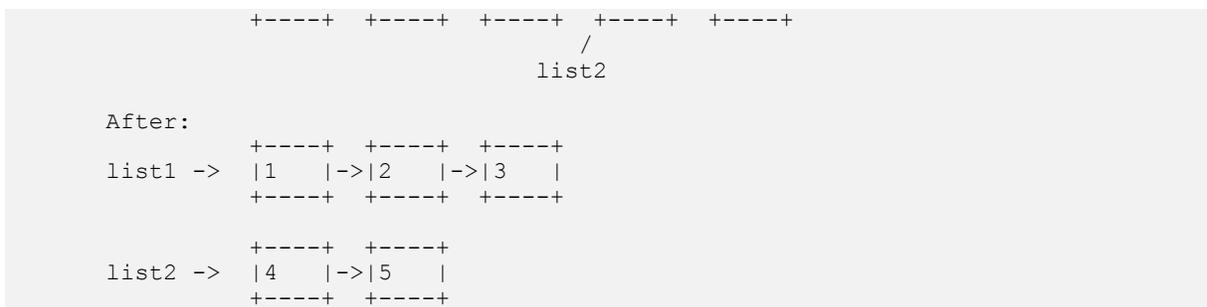
Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - if list2 is not part of list1.

Description



**Note:**

reference count of list2 will always be 1 after unchaining

2.4.3. DOCA Buffer Array

DOCA 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.

`typedef uint64_t doca_dpa_dev_buf_arr_t`

Type representing a `doca_buf_arr` handle on the DPA.

`DOCA_EXPERIMENTAL doca_error_t doca_buf_arr_create` `(size_t num_elem, doca_buf_arr **buf_arr)`

Allocates a `doca_buf_arr`.

Parameters

num_elem

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

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_EXPERIMENTAL 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.

Description

Destroy implicitly stops the buf array.

DOCA_EXPERIMENTAL doca_error_t doca_buf_arr_get_dpa_handle (const doca_buf_arr *buf_arr, doca_dpa_dev_buf_arr_t *dpa_buf_arr)

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

Parameters

buf_arr

The doca_buf_arr

dpa_buf_arr

A pointer to the handle in the dpa memory space

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_EXPERIMENTAL 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

A pointer to the handle in the gpu memory space

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_EXPERIMENTAL doca_error_t
doca_buf_arr_set_params (doca_buf_arr *buf_arr,
doca_mmap *mmap, size_t elem_size, uint64_t start_offset)
```

Sets the buf array params.

Parameters

buf_arr

The doca_buf_arr

mmap

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

elem_size

Size in bytes of a single element (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_EXPERIMENTAL doca_error_t  
doca_buf_arr_set_target_dpa (doca_buf_arr *buf_arr,  
doca_dpa *dpa_handler)
```

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

Parameters

buf_arr

The doca_buf_arr

dpa_handler

The dpa 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_EXPERIMENTAL 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_EXPERIMENTAL 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_NO_MEMORY -

Description



Note:

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

DOCA_EXPERIMENTAL 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.

Description



Note:

Stop does not have to be called before destroy (which implicitly stops the buf array).

2.4.4. DOCA Buffer Inventory

DOCA 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_STABLE 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_BAD_STATE - if `src_buf` mmap or input inventory unstarted/stopped.
- ▶ DOCA_ERROR_NOT_PERMITTED - if `src_buf` is part of a list and it isn't its head.
- ▶ DOCA_ERROR_NO_MEMORY - if cannot alloc new `doca_buf` from the given inventory.

Description

Call `doca_buf_dec_refcount` to return the buffer to the inventory (until ref count == 0).

```
doca_error_t doca_buf_inventory_buf_get_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.

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_NO_MEMORY - if doca_buf_inventory is empty.

Description

Call doca_buf_dec_refcount to return the buffer to the inventory (until ref count == 0).

```
DOCA_STABLE doca_error_t
doca_buf_inventory_buf_get_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.

len

The length in bytes of the buffer.

data

The start address of the data inside the buffer.

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

Call doca_buf_dec_refcount to return the buffer to the inventory (until ref count == 0).

**Note:**

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

```
doca_error_t doca_buf_inventory_buf_get_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.

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_NO_MEMORY - if doca_buf_inventory is empty.

Description

Call doca_buf_dec_refcount to return the buffer to the inventory (until ref count == 0).

DOCA_STABLE doca_error_t doca_buf_inventory_create
(size_t num_elements, doca_buf_inventory **inventory)

Allocates buffer inventory with default/unset attributes.

Parameters**num_elements**

Initial number of elements in the inventory.

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_STABLE 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_IN_USE - 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. Destroy implicitly stops the buf inventory. Call `doca_buf_dec_refcount` to return a buffer to the inventory (until ref count == 0).

```
DOCA_EXPERIMENTAL doca_error_t
doca_buf_inventory_expand (doca_buf_inventory *inventory,
uint32_t num_elements)
```

Expand the inventory.

Parameters

inventory

Inventory to expand

num_elements

Number of `doca_bufs` to add to the inventory

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_BAD_STATE - inventory is not started.

Description

Add more `doca_bufs` to the inventory.

DOCA_STABLE doca_error_t
 doca_buf_inventory_get_num_elements (const
 doca_buf_inventory *inventory, uint32_t *num_of_elements)

Read the total number of elements in a DOCA Buffer Inventory.

Parameters

inventory

The DOCA Buffer 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_STABLE doca_error_t
 doca_buf_inventory_get_num_free_elements
 (const doca_buf_inventory *inventory, uint32_t
 *num_of_free_elements)

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

Parameters

inventory

The DOCA Buffer 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_STABLE doca_error_t
doca_buf_inventory_get_user_data (const
doca_buf_inventory *inventory, doca_data *user_data)
```

Get the user_data of a DOCA Buffer Inventory.

Parameters

inventory

The DOCA Buffer Inventory.

user_data

The user_data of inventory if set, otherwise 0.

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_STABLE doca_error_t
doca_buf_inventory_set_user_data (doca_buf_inventory
*inventory, doca_data user_data)
```

Set user_data for a DOCA Buffer Inventory.

Parameters

inventory

The DOCA Buffer Inventory.

user_data

The user_data to set for inventory.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_BAD_STATE - if inventory is un-started/stopped.

DOCA_STABLE 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_get_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_STABLE 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.

buf

Pointer to the allocated `doca_buf`.

Description

Call `doca_buf_dec_refcount` to return the buffer to the pool (until ref count == 0).

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 `buf_pool` is un-started/stopped.
- ▶ DOCA_ERROR_EMPTY - if the `buf_pool` is empty (all `doca_bufs` are already allocated).

```
DOCA_STABLE doca_error_t doca_buf_pool_create (size_t
num_elements, size_t element_size, const doca_mmap
*mmap, doca_buf_pool **buf_pool)
```

Allocates a buffer pool and sets it with `doca_buf` objects.

Parameters**num_elements**

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

element_size

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

mmap

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

buf_pool

The newly created DOCA `buf_pool`.

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_pool`.

```
DOCA_STABLE doca_error_t doca_buf_pool_destroy
(doca_buf_pool *buf_pool)
```

Destroy a buffer pool structure.

Parameters**buf_pool**

The DOCA `buf_pool` 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 `buf_pool`.

Description

Destroy implicitly stops the `buf` pool.



Note:

Before Calling this method, all allocated `doca_bufs` should be returned back to the buffer pool. Call `doca_buf_dec_refcount` to return a buffer to the pool (until ref count == 0).

DOCA_STABLE `doca_error_t` `doca_buf_pool_get_element_alignment` (`const` `doca_buf_pool *buf_pool, size_t *element_alignment`)

Get the element alignment of a DOCA buffer pool.

Parameters

buf_pool

The DOCA `buf_pool`.

element_alignment

The element alignment of `buf_pool`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description



Note:

- Unless set with [doca_buf_pool_set_element_alignment\(\)](#), element alignment is 1 by default (meaning no alignment).

```
DOCA_STABLE doca_error_t
doca_buf_pool_get_num_elements (const doca_buf_pool
*buf_pool, uint32_t *num_of_elements)
```

Get the number of elements that was set in the creation of a DOCA buffer pool.

Parameters

buf_pool

The DOCA buf_pool.

num_of_elements

The number of elements that was set in the creation of buf_pool.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

```
DOCA_STABLE doca_error_t
doca_buf_pool_get_num_free_elements (const
doca_buf_pool *buf_pool, uint32_t *num_of_free_elements)
```

Get the total number of free elements available for allocation in a DOCA buffer pool.

Parameters

buf_pool

The DOCA buf_pool.

num_of_free_elements

The total number of free elements in buf_pool.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

```
DOCA_STABLE doca_error_t doca_buf_pool_get_user_data
(const doca_buf_pool *buf_pool, doca_data *user_data)
```

Get the user_data of a DOCA buffer pool.

Parameters

buf_pool

The DOCA buf_pool.

user_data

The user_data of buf_pool if set, otherwise 0.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description**Note:**

- Unless set with [doca_buf_pool_set_user_data\(\)](#), user data is 0 by default.

DOCA_STABLE doca_error_t
doca_buf_pool_set_element_alignment (doca_buf_pool
***buf_pool, size_t element_alignment)**

Set an alignment for each element in a DOCA buffer pool.

Parameters**buf_pool**

The DOCA buf_pool.

element_alignment

The element alignment to set for buf_pool (minimal value is 1, must be a power of 2).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_BAD_STATE - if buf_pool is started.

DOCA_STABLE doca_error_t doca_buf_pool_set_user_data
(doca_buf_pool *buf_pool, doca_data user_data)

Set user_data for a DOCA buffer pool.

Parameters**buf_pool**

The DOCA buf_pool.

user_data

The user_data to set for buf_pool.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_BAD_STATE - if buf_pool is started.

DOCA_STABLE doca_error_t doca_buf_pool_start (doca_buf_pool *buf_pool)

Start a DOCA buffer pool.

Parameters

buf_pool

The DOCA buf_pool 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_INITIALIZATION - if the mmap's memory range is smaller than the required size according to the buf_pool's properties.
- ▶ DOCA_ERROR_BAD_STATE - if the mmap with which buf_pool was created is not started.

Description

This method enables the allocation of doca_bufs using [doca_buf_pool_buf_alloc\(\)](#). Before calling this method, the mmap with which the buffer pool was created must be started.

The following become possible only after start:

- ▶ Allocating doca_bufs using [doca_buf_pool_buf_alloc\(\)](#).

The following are NOT possible while buf_pool is started:

- ▶ Setting properties of the buffer pool with [doca_buf_pool_set_*](#).

DOCA_STABLE doca_error_t doca_buf_pool_stop (doca_buf_pool *buf_pool)

Stop a started DOCA buffer pool.

Parameters

buf_pool

The DOCA buf_pool 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.
- ▶ DOCA_ERROR_IN_USE - if not all allocated `doca_bufs` had been returned to `buf_pool`.

Description

This method disables the allocation of `doca_bufs`, and re-enables Setting properties of the buffer pool with `doca_buf_pool_set_*`. Before Calling this method, all allocated `doca_bufs` should be returned back to the buffer pool. Stop does not have to be called before destroy (that implicitly stops the buf pool).

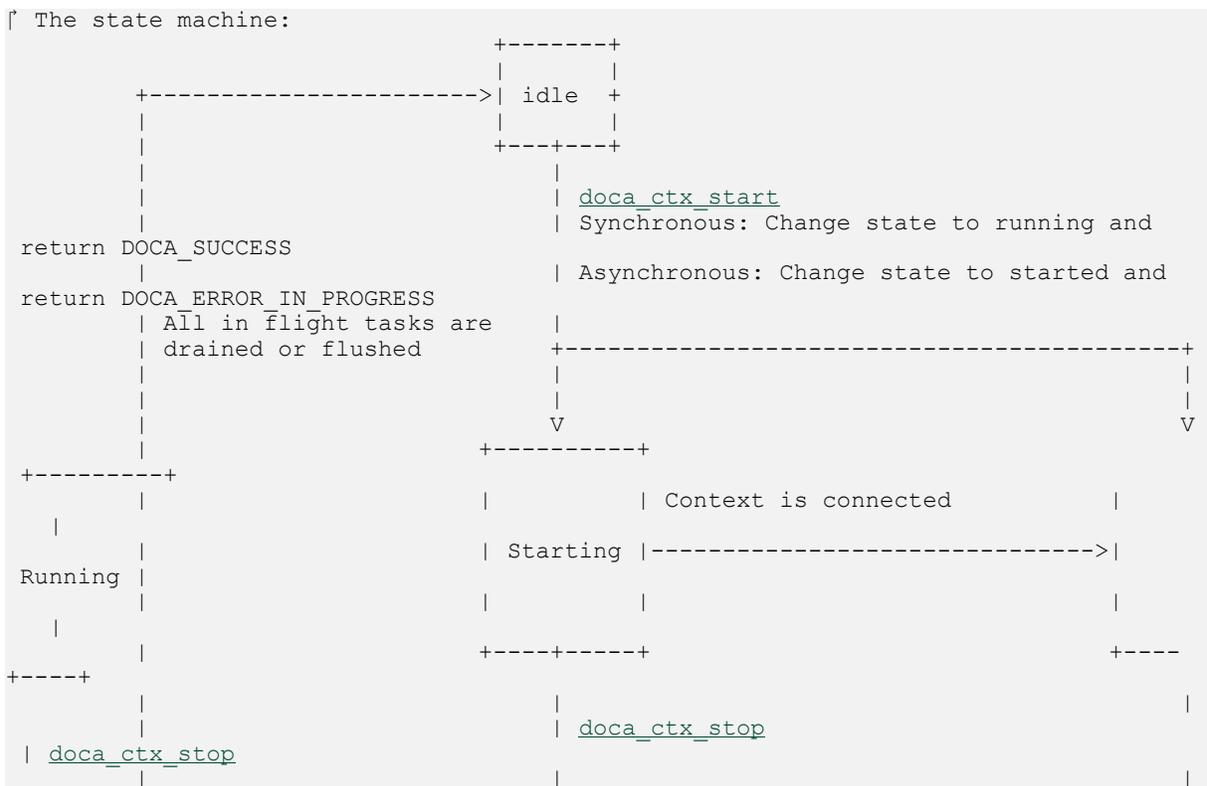
2.4.6. DOCA Context

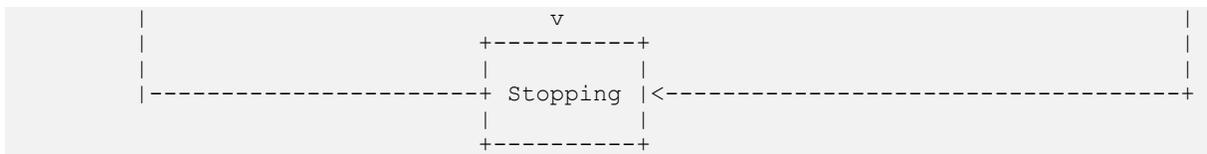
DOCA Core

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 tasks that manipulate data.

enum `doca_ctx_states`

This enum defines the states of a context.





Values

DOCA_CTX_STATE_IDLE = 0

ctx is created Resources are not allocated, ctx can not allocate tasks, submit tasks, allocate events or register events.

DOCA_CTX_STATE_STARTING = 1

doca_ctx_start called, context start is asynchronous. Resources are allocated, ctx can not allocate tasks, submit tasks, allocate events or register events.

DOCA_CTX_STATE_RUNNING = 2

doca_ctx_start called (ctx start is synchronous) or ctx connection is completed. Resources are allocated, ctx can allocate tasks, submit tasks, allocate events and register events.

DOCA_CTX_STATE_STOPPING = 3

```
typedef (*doca_ctx_state_changed_callback_t) (const union
doca_data user_data, doca_ctx* ctx, enum doca_ctx_states
prev_state, enum doca_ctx_states next_state)
```

Function to execute on context state change.

This function is called when a context state is changed.

See also:

[doca_ctx_set_user_data](#)

```
DOCA_EXPERIMENTAL void doca_ctx_flush_tasks (doca_ctx
*ctx)
```

Flushes tasks that were not flushed during submit.

Parameters

ctx

The DOCA context to flush. MUST NOT BE IDLE

Description

In case the DOCA_TASK_SUBMIT_FLAG_FLUSH was not provided during doca_task_submit_ex() this method can be used to flush the inflight tasks without the need to submit an additional task.

```
DOCA_STABLE doca_error_t
doca_ctx_get_num_inflight_tasks (const doca_ctx *ctx,
size_t *num_inflight_tasks)
```

Get number of in flight tasks in a doca context.

Parameters

ctx

Context to query

num_inflight_tasks

Total number of in flight tasks in the context

Returns

DOCA_SUCCESS Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This method retrieves the number of in flight tasks in a doca context

```
DOCA_STABLE doca_error_t doca_ctx_get_state (const
doca_ctx *ctx, doca_ctx_states **state)
```

Get context state.

Parameters

ctx

doca_ctx to get the state from

state

Current context state

Returns

DOCA_SUCCESS Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This method retrieves the context state

```
DOCA_STABLE doca_error_t doca_ctx_get_user_data (const
doca_ctx *ctx, doca_data *user_data)
```

get user data from context

Parameters

ctx

doca_ctx to get the user data from

user_data

user data to get

Returns

DOCA_SUCCESS Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This method retrieves user data from a context (previously set using doca_ctx_set_user_data).

```
DOCA_EXPERIMENTAL doca_error_t
doca_ctx_set_datapath_on_dpa (doca_ctx *ctx, doca_dpa
*dpa_dev)
```

This function binds the DOCA context to a dpa device.

Parameters

ctx

The library instance.

dpa_dev

A pointer to a doca_dpa 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_EXPERIMENTAL 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_STABLE doca_error_t
doca_ctx_set_state_changed_cb (doca_ctx *ctx,
doca_ctx_state_changed_callback_t cb)
```

Set state changed callback.

Parameters

ctx

doca_ctx to set the callback to

cb

doca_ctx_state_changed_callback_t

Returns

DOCA_SUCCESS Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This method sets state changed callback that is invoked every time that a context state is changed

DOCA_STABLE doca_error_t doca_ctx_set_user_data (doca_ctx *ctx, doca_data user_data)

set user data to context

Parameters

ctx

doca_ctx to set the user data to

user_data

doca_data to set to the context

Returns

DOCA_SUCCESS Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

This method sets a user data to a context. The user data is used as a parameter in `doca_ctx_state_changed_callback_t`

DOCA_STABLE 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_NOT_CONNECTED - ctx is not connected to a PE and data path on gpu or dpa was not set.
- ▶ 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:

- ▶ Submitting a task using `doca_task_submit()`

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

- ▶ Changing CTX properties
- ▶ Binding gpu device to CTX using `doca_ctx_set_datapath_on_gpu()`
- ▶ Binding dpa device to CTX using `doca_ctx_set_datapath_on_dpa()`

DOCA_STABLE `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_IN_PROGRESS - some tasks are still in progress. CTX will move to stopping state and a state changed callback shall be invoked when context is fully stopped.
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_CONNECTED - ctx is not connected to a PE and data path on gpu or dpa was not set.
- ▶ 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()`.

2.4.7. DOCA Device

DOCA Core

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

enum doca_devinfo_rep_filter

Representor device filter by flavor

Multiple options possible but some are mutually exclusive.

Values

DOCA_DEVINFO_REP_FILTER_ALL = 0

DOCA_DEVINFO_REP_FILTER_NET = 1<<1

DOCA_DEVINFO_REP_FILTER_EMULATED = 1<<2

DOCA_STABLE 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_rdma_bridge_open_dev_from_pd().

DOCA_STABLE 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.

Description

Closes device or decrements its refcount by One. In case the same device was opened multiple times, then only the last call to close will attempt to destroy device.

DOCA_STABLE `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

Opens device or increments its refcount by One. The device can later be used by other libraries. For every call to [doca_dev_open\(\)](#) there should be a call to [doca_dev_close\(\)](#).

DOCA_STABLE `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_STABLE 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_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()

DOCA_STABLE 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_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()

```
DOCA_STABLE doca_error_t
doca_devinfo_cap_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_EXPERIMENTAL doca_error_t
doca_devinfo_cap_is_notification_moderation_supported
(const doca_devinfo *devinfo, uint8_t
*is_notification_moderation_supported)
```

Check if notification moderation is supported for a device.

Parameters

devinfo

The device to query.

is_notification_moderation_supported

1 if the device supports notification moderation

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 notification moderation supported type: uint8_t*.

DOCA_STABLE doca_error_t doca_devinfo_create_list (doca_devinfo dev_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_destroy_list\(\)](#)

DOCA_STABLE doca_error_t doca_devinfo_destroy_list (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_EXPERIMENTAL `doca_error_t`
`doca_devinfo_get_active_rate (const doca_devinfo *devinfo,`
`uint64_t *active_rate)`

Get the active rate of a DOCA devinfo.

Parameters

devinfo

The device to query.

active_rate

The active rate of the given port on the device. Given in units of bits/s.

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 rate.

DOCA_STABLE `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.

Description

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

`DOCA_STABLE 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_STABLE 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_NOT_FOUND - no IPv4 address was assigned
- ▶ 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_STABLE 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_STABLE 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_STABLE 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_STABLE doca_error_t
doca_devinfo_get_pci_addr_str (const doca_devinfo
*devinfo, char *pci_addr_str)
```

Get the PCI address of a DOCA devinfo.

Parameters

devinfo

The device to query.

pci_addr_str

The PCI address of devinfo, should be of size DOCA_DEVINFO_PCI_ADDR_SIZE at least.

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 PCI address from the OS

Description

The PCI address string format is "Domain:Bus:Device.Function", such that each value is represented by HEX digits, e.g., "0000:3a:00.0"

```
DOCA_STABLE doca_error_t
doca_devinfo_is_equal_pci_addr (const doca_devinfo
*devinfo, const char *pci_addr_str, uint8_t *is_equal)
```

Check if a PCI address belongs to a DOCA devinfo.

Parameters

devinfo

The device to query.

pci_addr_str

"Domain:Bus:Device.Function", e.g., "0000:3a:00.0" (size DOCA_DEVINFO_PCI_ADDR_SIZE including a null terminator). "Bus:Device.Function", e.g., "3a:00.0" (size DOCA_DEVINFO_PCI_BDF_SIZE including a null terminator).

The PCI address to check, should be as one of the following formats:

is_equal

1 if pci_addr_str belongs to devinfo, 0 otherwise. In case of an error, no certain value is guaranteed.

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 actual PCI address from the OS for comparison.
- ▶ "Domain:Bus:Device.Function", e.g., "0000:3a:00.0" (size DOCA_DEVINFO_PCI_ADDR_SIZE including a null terminator).
- ▶ "Bus:Device.Function", e.g., "3a:00.0" (size DOCA_DEVINFO_PCI_BDF_SIZE including a null terminator).

DOCA_STABLE doca_error_t doca_devinfo_rep_cap_is_filter_all_supported (const doca_devinfo *devinfo, uint8_t *filter_all_supported)

Get the representor devices discovery capability of the device.

Parameters**devinfo**

The device to query.

filter_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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()

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_create_list\(\)](#). true - device can be used with the remote list create API with filter DOCA_DEVINFO_REP_FILTER_ALL. false - providing DOCA_DEVINFO_REP_FILTER_ALL is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

DOCA_STABLE doca_error_t
doca_devinfo_rep_cap_is_filter_emulated_supported (const doca_devinfo *devinfo, uint8_t *filter_emulated_supported)

Get the remote emulated device discovery capability of the device.

Parameters

devinfo

The device to query.

filter_emulated_supported

1 if the list emulated 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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by `doca_rdma_bridge_open_dev_from_pd()`

Description

Get uint8_t value defining if the device can be used to create list of emulated representor devices. See [doca_devinfo_rep_create_list\(\)](#). true - device can be used with the remote list create API with filter DOCA_DEVINFO_REP_FILTER_EMULATED. false - providing DOCA_DEVINFO_REP_FILTER_EMULATED is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

DOCA_STABLE doca_error_t
doca_devinfo_rep_cap_is_filter_net_supported (const doca_devinfo *devinfo, uint8_t *filter_net_supported)

Get the remote net discovery capability of the device.

Parameters

devinfo

The device to query.

filter_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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by `doca_rdma_bridge_open_dev_from_pd()`

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_STABLE `doca_error_t doca_devinfo_rep_create_list(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_devinfo_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_DRIVER - Failed to query driver.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by `doca_rdma_bridge_open_dev_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_destroy_list\(\)](#)

DOCA_STABLE doca_error_t doca_devinfo_rep_destroy_list (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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()

Description

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

DOCA_STABLE doca_error_t doca_devinfo_rep_get_is_hotplug (const doca_devinfo_rep *devinfo_rep, uint8_t *is_hotplug)

Query whether the representor device is a hotplugged device.

Parameters

devinfo_rep

representor device info

is_hotplug

1 if the representor device is a hotplugged device. 0 if representor device is statically plugged.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

DOCA_STABLE doca_error_t doca_devinfo_rep_get_pci_addr_str (const doca_devinfo_rep *devinfo_rep, char *pci_addr_str)

Get the PCI address of a DOCA devinfo_rep.

Parameters**devinfo_rep**

The device to query.

pci_addr_str

The PCI address of devinfo_rep, should be of size DOCA_DEVINFO_REP_PCI_ADDR_SIZE at least.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()
- ▶ DOCA_ERROR_NO_MEMORY - not enough memory to generate the stringed PCI address.
- ▶ DOCA_ERROR_UNEXPECTED - an unexpected error occurred.

Description

The PCI address string format is "Domain:Bus:Device.Function", such that each value is represented by HEX digits, e.g., "0000:3a:00.0".

```
DOCA_STABLE doca_error_t
doca_devinfo_rep_get_pci_func_type (const
doca_devinfo_rep *devinfo_rep, enum 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.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by doca_rdma_bridge_open_dev_from_pd()

Description

The pci function type: enum doca_pci_func_type.

```
DOCA_STABLE 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 ('\0').

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by `doca_rdma_bridge_open_dev_from_pd()`

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_STABLE doca_error_t
doca_devinfo_rep_is_equal_pci_addr (const
doca_devinfo_rep *devinfo_rep, const char *pci_addr_str,
uint8_t *is_equal)
```

Check if a PCI address belongs to a DOCA `devinfo_rep`.

Parameters

devinfo_rep

The representor of device to query.

pci_addr_str

"Domain:Bus:Device.Function", e.g., "0000:3a:00.0" (size DOCA_DEVINFO_PCI_ADDR_SIZE including a null terminator). "Bus:Device.Function", e.g., "3a:00.0" (size DOCA_DEVINFO_PCI_BDF_SIZE including a null terminator).

The PCI address to check, should be as one of the following formats:

is_equal

1 if `pci_addr_str` belongs to `devinfo_rep`, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices, or dev was created by `doca_rdma_bridge_open_dev_from_pd()`
- ▶ DOCA_ERROR_NO_MEMORY - not enough memory to generate `devinfo_rep` PCI address for comparison.
- ▶ DOCA_ERROR_UNEXPECTED - an unexpected error occurred.
- ▶ "Domain:Bus:Device.Function", e.g., "0000:3a:00.0" (size DOCA_DEVINFO_PCI_ADDR_SIZE including a null terminator).
- ▶ "Bus:Device.Function", e.g., "3a:00.0" (size DOCA_DEVINFO_PCI_BDF_SIZE including a null terminator).

#define DOCA_DEVINFO_IBDEV_NAME_SIZE 64

Buffer size to hold Infiniband/RoCE device name. Including a null terminator.

```
#define DOCA_DEVINFO_IFACE_NAME_SIZE 256
```

Buffer size to hold network interface name. Including a null terminator.

```
#define DOCA_DEVINFO_IPV4_ADDR_SIZE 4
```

Length of IPv4 address.

```
#define DOCA_DEVINFO_IPV6_ADDR_SIZE 16
```

Length of IPv6 address.

```
#define DOCA_DEVINFO_MAC_ADDR_SIZE 6
```

Length of MAC address.

```
#define DOCA_DEVINFO_PCI_ADDR_SIZE 13
```

Buffer size to hold PCI BDF format: "XXX:XX:XX.X". Including a null terminator.

```
#define DOCA_DEVINFO_PCI_BDF_SIZE 8
```

Buffer size to hold PCI BDF format: "XX:XX.X". Including a null terminator.

```
#define DOCA_DEVINFO_REP_PCI_ADDR_SIZE 13
```

Buffer size to hold PCI BDF format: "XXX:XX:XX.X". Including a null terminator.

```
#define DOCA_DEVINFO_REP_PCI_BDF_SIZE 8
```

Buffer size to hold PCI BDF format: "XX:XX.X". Including a null terminator.

```
#define DOCA_DEVINFO_REP_VUID_SIZE 128
```

Buffer size to hold VUID. Including a null terminator.

```
#define DOCA_DEVINFO_VUID_SIZE 128
```

Buffer size to hold VUID. Including a null terminator.

2.5. DOCA Comch

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.

DOCA Comch Consumer

DOCA Comch MsgQ

DOCA Comch Producer

enum doca_comch_counter

Available counters for connection statistics query

Values

DOCA_COMCH_COUNTER_SENT_MESSAGES = 1

DOCA_COMCH_COUNTER_SENT_BYTES = 2

DOCA_COMCH_COUNTER_RECV_MESSAGES = 3

DOCA_COMCH_COUNTER_RECV_BYTES = 4

typedef

```
(*doca_comch_event_connection_status_changed_cb_t)
(doca_comch_event_connection_status_changed*
event, doca_comch_connection* comch_connection,
uint8_t change_successful)
```

Function executed on a doca_comch connection event.

The implementation can assume these values are not NULL.

```
typedef (*doca_comch_event_consumer_cb_t)
(doca_comch_event_consumer* event,
doca_comch_connection* comch_connection,
uint32_t id)
```

Function executed on a doca_comch consumer event.

The implementation can assume these values are not NULL.

```
typedef (*doca_comch_event_msg_rcv_cb_t)
(doca_comch_event_msg_rcv* event,
uint8_t* rcv_buffer, uint32_t msg_len,
doca_comch_connection* comch_connection)
```

Function executed on a doca_comch receive message event.

The implementation can assume these values are not NULL.

```
typedef (*doca_comch_task_send_completion_cb_t)
(doca_comch_task_send* task, union doca_data
task_user_data, union doca_data ctx_user_data)
```

Function executed on doca_comch send task completion. Used for both task success and failure.

The implementation can assume this value is not NULL.

```
DOCA_STABLE doca_error_t
doca_comch_cap_client_is_supported (const
doca_devinfo *devinfo)
```

Parameters

devinfo

The DOCA device information.

Returns

DOCA_SUCCESS - in case device can run as a client. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo can not implement a cc client

Description

Check if given device is capable of running a comm channel client.

DOCA_STABLE doca_error_t
 doca_comch_cap_get_max_clients (const
 doca_devinfo *devinfo, uint32_t *num_clients)

Parameters

devinfo

devinfo to query the capability for.

num_clients

The number of clients that can be connected to a single doca_comch server.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximal number of clients that can be connected to a single doca_comch server.

DOCA_STABLE doca_error_t
 doca_comch_cap_get_max_msg_size (const
 doca_devinfo *devinfo, uint32_t *size)

Parameters

devinfo

devinfo to query the capability for.

size

The maximum size of a message available on any cc instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximum message size that can be used on any comm channel instance.

```
DOCA_STABLE doca_error_t
doca_comch_cap_get_max_name_len (const
doca_devinfo *devinfo, uint32_t *max_name_len)
```

Parameters

devinfo

devinfo to query the capability for.

max_name_len

The cc max name length, including the terminating null byte ('\0').

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximum name length that can be used in a cc instance.

```
DOCA_STABLE doca_error_t
doca_comch_cap_get_max_recv_queue_size (const
doca_devinfo *devinfo, uint32_t *size)
```

Parameters

devinfo

devinfo to query the capability for.

size

The maximal recv queue size supported on any cc instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximal recv queue size that can be used on any comm channel instance.

```
DOCA_STABLE doca_error_t
doca_comch_cap_get_max_send_tasks (const
doca_devinfo *devinfo, uint32_t *max_send_tasks)
```

Parameters

devinfo

devinfo to query the capability for.

max_send_tasks

The maximal supported number of send tasks for any cc instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the maximal send tasks num that can be used on any cc instance.

```
DOCA_STABLE doca_error_t
doca_comch_cap_server_is_supported (const
doca_devinfo *devinfo)
```

Parameters

devinfo

The DOCA device information.

Returns

DOCA_SUCCESS - in case device can run as a server. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo can not implement a cc server.

Description

Check if given device is capable of running a comm channel server.

```
DOCA_STABLE doca_ctx *doca_comch_client_as_ctx
(doca_comch_client *comch_client)
```

Parameters

comch_client

DOCA Comch client instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert `doca_comch` instance into a generalized context for use with DOCA core objects.

```
DOCA_STABLE doca_error_t
doca_comch_client_create (doca_dev *dev, const char
*name, doca_comch_client **comch_client)
```

Parameters

dev

Device to use in DOCA Comch client instance.

name

Identifier for the server the client will connect to. Max length, including terminating '\0', is obtained by [doca_comch_cap_get_max_name_len\(\)](#).

comch_client

Pointer to pointer to be set to created `doca_comch` client instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - one or more of the arguments is null.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc `doca_comch`.
- ▶ DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Create a DOCA Comch client instance.

```
DOCA_STABLE doca_error_t
doca_comch_client_destroy (doca_comch_client
*comch_client)
```

Parameters

comch_client

DOCA Comch client instance to be destroyed.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - cc argument is a NULL pointer.
- ▶ DOCA_ERROR_IN_USE - Unable to gain exclusive access to the cc instance.

Description

Destroy a DOCA Comch client instance.

```
DOCA_STABLE doca_error_t
doca_comch_client_event_consumer_register
(doca_comch_client *comch_client,
doca_comch_event_consumer_cb_t
new_consumer_event_cb,
doca_comch_event_consumer_cb_t
expired_consumer_event_cb)
```

Configure the doca_comch callback for for receiving consumer events on client context.

Parameters

comch_client

Pointer to doca_comch_client instance.

new_consumer_event_cb

Consumer event callback on creation of a new consumer.

expired_consumer_event_cb

Consumer event callback on when a consumer has expired.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case one of the arguments is NULL.
- ▶ DOCA_ERROR_BAD_STATE - `doca_comch_servcie` context state is not idle.

```
DOCA_STABLE doca_error_t
doca_comch_client_event_msg_rcv_register
(doca_comch_client *comch_client,
doca_comch_event_msg_rcv_cb_t rcv_event_cb)
```

Configure the `doca_comch` rcv event callback for client context.

Parameters

comch_client

Pointer to `doca_comch_client` instance.

rcv_event_cb

Rcv event callback.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case one of the arguments is NULL.
- ▶ DOCA_ERROR_BAD_STATE - `doca_comch` context state is not idle.

```
DOCA_STABLE doca_comch_client
*doca_comch_client_get_client_ctx (const
doca_comch_connection *connection)
```

Parameters

connection

DOCA Comch connection instance.

Returns

`doca_comch_client` object on success. NULL if the connection is related to a server context.

Description

Get the `doca_comch_client` context from a given connection.

```
DOCA_STABLE doca_error_t
doca_comch_client_get_connection
(const doca_comch_client *comch_client,
 doca_comch_connection **connection)
```

Parameters

comch_client

DOCA Comch client instance.

connection

The connection object associated with the client.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is not started.

Description

Get the connection object associated with the client ctx. Can only be called after starting the ctx.

```
DOCA_STABLE doca_error_t
doca_comch_client_get_device (const
 doca_comch_client *comch_client, doca_dev **dev)
```

Parameters

comch_client

DOCA Comch client instance.

dev

Current device used in the doca_comch instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Get the doca device property of the associated doca_comch instance.

```
DOCA_STABLE doca_error_t
doca_comch_client_get_max_msg_size (const
doca_comch_client *comch_client, uint32_t *size)
```

Parameters

comch_client

DOCA Comch client instance.

size

The maximum size of a message for the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Get the maximum message size that can be sent on the comm channel instance.

```
DOCA_STABLE doca_error_t
doca_comch_client_get_recv_queue_size (const
doca_comch_client *comch_client, uint32_t *size)
```

Parameters

comch_client

DOCA Comch client instance.

size

The recv queue size for the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the recv queue size property set on the `doca_comch` instance.

```
DOCA_STABLE doca_error_t
doca_comch_client_set_max_msg_size
(doca_comch_client *comch_client, uint32_t size)
```

Parameters

comch_client

DOCA Comch client instance.

size

The maximum size of a message to set for the instance. Can be queried with [doca_comch_cap_get_max_msg_size\(\)](#).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the maximum message size property for the `doca_comch` instance. If not called, a default value will be used and can be queried using [doca_comch_client_get_max_msg_size\(\)](#).

```
DOCA_STABLE doca_error_t
doca_comch_client_set_recv_queue_size
(doca_comch_client *comch_client, uint32_t size)
```

Parameters

comch_client

DOCA Comch client instance.

size

The recv queue size to set for the instance. Limit can be queried with [doca_comch_cap_get_max_recv_queue_size\(\)](#).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the recv queue size property for the `doca_comch` instance. If not called, a default value will be used and can be queried using [doca_comch_client_get_recv_queue_size\(\)](#).

```
DOCA_STABLE doca_error_t
doca_comch_client_task_send_alloc_init
(doca_comch_client *comch_client,
doca_comch_connection *peer, const void *msg,
uint32_t len, doca_comch_task_send **task)
```

Parameters

comch_client

The `doca_comch_client` instance.

peer

Connected endpoint to send the message to.

msg

Message or data to send to associated peer.

len

Length of the message to send.

task

Pointer to a `doca_comch_send_task` instance populated with input parameters.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NO_MEMORY - no available tasks to allocate.

Description

Allocate and initialize a `doca_comch_client` send task.

```
DOCA_STABLE doca_error_t  
doca_comch_client_task_send_set_conf  
(doca_comch_client *comch_client,  
doca_comch_task_send_completion_cb_t  
task_completion_cb,  
doca_comch_task_send_completion_cb_t  
task_error_cb, uint32_t num_send_tasks)
```

Parameters

comch_client

The `doca_comch_client` instance.

task_completion_cb

Send task completion callback.

task_error_cb

Send task error callback.

num_send_tasks

Number of send tasks to create.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Configure the `doca_comch_client` send task callback and parameters.

```
DOCA_STABLE doca_error_t
doca_comch_connection_get_counter (const
doca_comch_connection *comch_connection,
doca_comch_counter_counter_type, uint64_t
*counter_value)
```

get statistics counter for a given comch_connection

Parameters

comch_connection

Pointer to comch_connection to query statistics for.

counter_type

Which statistics counter should be queried.

counter_value

Will contain the value for the counter on the given comch_connection.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL or if the counter is not valid.

Description

This function will return statistics for a given comch_connection, updated to the last time [doca_comch_connection_update_info\(\)](#) was called.

```
DOCA_STABLE doca_data
doca_comch_connection_get_user_data (const
doca_comch_connection *connection)
```

Parameters

connection

DOCA Comch connection instance.

Returns

User data for the given connection.

Description

Get the user data from a given connection.

```
DOCA_STABLE doca_error_t
doca_comch_connection_set_user_data
(doca_comch_connection *connection, doca_data
user_data)
```

Parameters

connection

DOCA Comch connection instance.

user_data

User data for the given connection.

Returns

DOCA_SUCCESS on success.

Description

Set the user data for a given connection.

```
DOCA_STABLE doca_error_t
doca_comch_connection_update_info
(doca_comch_connection *comch_connection)
```

update statistics for given comch_connection

Parameters

comch_connection

Pointer to comch_connection to update statistics in.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if comch_connection 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 connection information function to update the saved statistics.

```
DOCA_STABLE doca_ctx *doca_comch_server_as_ctx
(doca_comch_server *comch_server)
```

Parameters

comch_server

DOCA Comch server instance. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert `doca_comch_server` instance into a generalized context for use with DOCA core objects.

```
DOCA_STABLE doca_error_t
doca_comch_server_create (doca_dev *dev,
doca_dev_rep *repr, const char *name,
doca_comch_server **comch_server)
```

Parameters

dev

Device to use in DOCA Comch server instance.

repr

Representor device to use in CC server instance.

name

Identifier for server associated with instance. Must be NULL terminated. Max length, including terminating '\0', is obtained by `doca_comch_cap_get_max_name_len()`.

comch_server

Pointer to pointer to be set to created `doca_comch` server instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - one or more of the arguments is null.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc `doca_comch`.
- ▶ DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Create a DOCA Comch server instance.

```
DOCA_STABLE doca_error_t
doca_comch_server_destroy (doca_comch_server
*comch_server)
```

Parameters

comch_server

DOCA Comch server instance to be destroyed.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - cc argument is a NULL pointer.
- ▶ DOCA_ERROR_IN_USE - Unable to gain exclusive access to the cc instance.

Description

Destroy a DOCA Comch server instance.

```
DOCA_STABLE doca_error_t
doca_comch_server_disconnect (doca_comch_server
*comch_server, doca_comch_connection
*connection)
```

Parameters

comch_server

DOCA Comch server instance.

connection

Connection representing the remote peer to disconnect.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_IN_USE - if there are active consumers/producers for the connection.

- ▶ DOCA_ERROR_AGAIN - cannot send disconnection message due to full queue. Requires the app to call disconnect again later.

Description

Disconnect specific connection on `doca_comch_server`.

This function will notify the peer of the disconnection.

```
DOCA_STABLE doca_error_t
doca_comch_server_event_connection_status_changed_register(
doca_comch_server *comch_server,
doca_comch_event_connection_status_changed_cb_t
connect_event_cb,
doca_comch_event_connection_status_changed_cb_t
disconnect_event_cb)
```

Configure the `doca_comch` recv event callback for server context.

Parameters

comch_server

Pointer to `doca_comch_server` instance.

connect_event_cb

Callback for connect event.

disconnect_event_cb

Callback for disconnect event.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case one of the arguments is NULL.
- ▶ DOCA_ERROR_BAD_STATE - `doca_comch` context state is not idle.

```
DOCA_STABLE doca_error_t
doca_comch_server_event_consumer_register
(doca_comch_server *comch_server,
doca_comch_event_consumer_cb_t
new_consumer_event_cb,
doca_comch_event_consumer_cb_t
expired_consumer_event_cb)
```

Configure the `doca_comch` callback for for receiving consumer events on server context.

Parameters

comch_server

Pointer to `doca_comch_server` instance.

new_consumer_event_cb

Consumer event callback on creation of a new consumer.

expired_consumer_event_cb

Consumer event callback on when a consumer has expired.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case one of the arguments is NULL.
- ▶ DOCA_ERROR_BAD_STATE - `doca_comch_servcie` context state is not idle.

```
DOCA_STABLE doca_error_t
doca_comch_server_event_msg_rcv_register
(doca_comch_server *comch_server,
doca_comch_event_msg_rcv_cb_t rcv_event_cb)
```

Configure the `doca_comch` `rcv` event callback for server context.

Parameters

comch_server

Pointer to `doca_comch_server` instance.

rcv_event_cb

Recv event callback.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case one of the arguments is NULL.
- ▶ DOCA_ERROR_BAD_STATE - `doca_comch` context state is not idle.

```
DOCA_STABLE doca_error_t
doca_comch_server_get_device (const
doca_comch_server *comch_server, doca_dev **dev)
```

Parameters

comch_server

DOCA Comch server instance.

dev

Current device used in the `doca_comch` instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - `cc` instance is already active.

Description

Get the `doca` device property of the associated `doca_comch` instance.

```
DOCA_STABLE doca_error_t
doca_comch_server_get_device_rep (const
doca_comch_server *comch_server, doca_dev_rep
**rep)
```

Parameters

comch_server

DOCA Comch server instance.

rep

Current device representor used in the `doca_comch` server instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Get the device representor property of the associated doca_comch server instance.

```
DOCA_STABLE doca_error_t
doca_comch_server_get_max_msg_size (const
doca_comch_server *comch_server, uint32_t *size)
```

Parameters

comch_server

DOCA Comch server instance.

size

The maximum size of a message for the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Get the maximum message size that can be sent on the comm channel instance.

```
DOCA_STABLE doca_error_t
doca_comch_server_get_recv_queue_size (const
doca_comch_server *comch_server, uint32_t *size)
```

Parameters

comch_server

DOCA Comch server instance.

size

The recv queue size set for the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the recv queue size property set on the `doca_comch` instance.

```
DOCA_STABLE doca_comch_server
*doca_comch_server_get_server_ctx (const
doca_comch_connection *connection)
```

Parameters

connection

DOCA Comch connection instance.

Returns

`doca_comch_server` object on success. NULL if the connection is related to a client context.

Description

Get the `doca_comch_server` context from a given connection.

```
DOCA_STABLE doca_error_t
doca_comch_server_set_max_msg_size
(doca_comch_server *comch_server, uint32_t size)
```

Parameters

comch_server

DOCA Comch server instance.

size

The maximum size of a message to set for the instance. Can be queried with [doca_comch_cap_get_max_msg_size\(\)](#).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the maximum message size property for the `doca_comch` instance. If not called, a default value will be used and can be queried using [doca_comch_server_get_max_msg_size\(\)](#).

```
DOCA_STABLE doca_error_t
doca_comch_server_set_rcv_queue_size
(doca_comch_server *comch_server, uint32_t size)
```

Parameters

comch_server

DOCA Comch server instance.

size

The rcv queue size set for the instance. Can be queried with [doca_comch_cap_get_max_rcv_queue_size\(\)](#).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the rcv queue size property for the `doca_comch` instance. If not called, a default value will be used and can be queried using [doca_comch_server_get_rcv_queue_size\(\)](#).

```
DOCA_STABLE doca_error_t
doca_comch_server_task_send_alloc_init
(doca_comch_server *comch_server,
doca_comch_connection *peer, const void *msg,
uint32_t len, doca_comch_task_send **task)
```

Parameters

comch_server

The `doca_comch_server` instance.

peer

Connected endpoint to send the message to.

msg

Message or data to sent to associated peer.

len

Length of the message to send.

task

Pointer to a `doca_comch_send_task` instance populated with input parameters.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NO_MEMORY - no available tasks to allocate.

Description

Allocate and initialize a `doca_comch_server` send task.

```
DOCA_STABLE doca_error_t
doca_comch_server_task_send_set_conf
(doca_comch_server *comch_server,
 doca_comch_task_send_completion_cb_t
 task_completion_cb,
 doca_comch_task_send_completion_cb_t
 task_error_cb, uint32_t num_send_tasks)
```

Parameters**comch_server**

The `doca_comch_server` instance.

task_completion_cb

Send task completion callback.

task_error_cb

Send task error callback.

num_send_tasks

Number of send tasks to create.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Configure the `doca_comch_server` send task callback and parameters.

```
DOCA_STABLE doca_task
*doca_comch_task_send_as_task
(doca_comch_task_send *task)
```

Parameters

task

`Doca_comch_send_task` task to convert.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert a `doca_comch_send_task` task to `doca_task`.

2.5.1. DOCA Comch Consumer

DOCA Comch

DOCA Communication Channel Consumer offers an extension the `doca_comch` channel for accelerated data transfer between memory on the host and DPU in a FIFO format. An established `doca_comch` connection is required to negotiate the end points of the FIFO. A consumer object can then post buffers to a remote process that it wishes to receive data on. Completion of a consumer post receive message indicates that data has been populated from a remote producer. The inter-process communication runs over DMA/PCIe and does not affect network bandwidth.

typedef

```
(*doca_comch_consumer_task_post_rcv_completion_cb_t)
(doca_comch_consumer_task_post_rcv* task, union
doca_data task_user_data, union doca_data ctx_user_data)
```

Function executed on `doca_comch_consumer` post receive completion. Used for both task success and failure.

The implementation can assume this value is not NULL.

```
typedef uint64_t
doca_dpa_dev_comch_consumer_completion_t
```

DPA handle for DPA consumer completion context.

```
typedef uint64_t doca_dpa_dev_comch_consumer_t
```

DPA handle for DPA consumer.

```
DOCA_STABLE doca_ctx *doca_comch_consumer_as_ctx
(doca_comch_consumer *consumer)
```

Parameters

consumer

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

Returns

Non NULL upon success, NULL otherwise.

Description

Convert doca_comch_consumer instance into a generalized context for use with doca core objects.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_get_max_buf_list_len (const
doca_devinfo *devinfo, uint32_t *max_buf_list_len)
```

Parameters

devinfo

Devinfo to query the capability for.

max_buf_list_len

Maximum sized buffer list that can be received by the consumer.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - if consumer is not supported on device.

Description

Get the max length of `doca_buf` list that can be received by a `doca_comch_consumer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_get_max_buf_size (const
doca_devinfo *devinfo, uint32_t *max_buf_size)
```

Parameters

devinfo

Devinfo to query the capability for.

max_buf_size

Maximum sized buffer that can be received by the consumer.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - if consumer is not supported on device.

Description

Get the max size `doca_buf` that can be received by a `doca_comch_consumer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_get_max_consumers (const
doca_devinfo *devinfo, uint32_t *max_consumers)
```

Parameters

devinfo

Devinfo to query the capability for.

max_consumers

Maximum number of consumers that can be added to a `doca_comch_connection`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the max number of consumers that can be associated with a `doca_comch_connection`.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_get_max_imm_data_len
(const doca_devinfo *devinfo, uint32_t *max_imm_data_len)
```

Parameters

devinfo

Devinfo to query the capability for.

max_imm_data_len

Maximum length of immediate data consumers support.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the max length of immediate data supported by consumers.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_get_max_num_tasks (const
doca_devinfo *devinfo, uint32_t *max_num_tasks)
```

Parameters

devinfo

Devinfo to query the capability for.

max_num_tasks

The maximum number of tasks that can allocated by the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the max number of tasks supported by the device for a `doca_comch_consumer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_cap_is_supported (const
doca_devinfo *devinfo)
```

Parameters

devinfo

The DOCA device information.

Returns

DOCA_SUCCESS - in case device can implement a consumer. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo can not implement a consumer.

Description

Check if given device is capable of running a consumer.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_create
(doca_comch_consumer_completion **consumer_comp)
```

Allocate DOCA Comch consumer completion context on DPA.

Parameters

consumer_comp

The newly created DOCA Comch consumer completion context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_destroy
(doca_comch_consumer_completion *consumer_comp)
```

Destroy the DOCA Comch consumer completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to destroy.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

The associated dpa handle will be destroyed as well.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_get_dpa_handle
(doca_comch_consumer_completion *consumer_comp,
doca_dpa_dev_comch_consumer_completion_t
*consumer_comp_handle)
```

Get the DPA handle for the DOCA Comch consumer completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context previously created on DPA.

consumer_comp_handle

A pointer to the associated DPA handle in the dpa memory space.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_get_imm_data_len
(const doca_comch_consumer_completion
*consumer_comp, uint32_t *imm_data_len)
```

Parameters

consumer_comp

The DOCA Comch consumer completion context to query.

imm_data_len

Length of immediate data the consumer completion context is configured to support.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the length of immediate data supported by a consumer completion context.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_get_max_num_consumers
(const doca_comch_consumer_completion
*consumer_comp, uint32_t *max_num_consumers)
```

Get the maximal number of consumers that can be associated with the completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to query.

max_num_consumers

The maximal number of consumers that can be associated with the context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_get_max_num_rcv
(const doca_comch_consumer_completion
*consumer_comp, uint32_t *max_num_rcv)
```

Get the maximal number of receive operations across all consumers associated with the completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to query.

max_num_rcv

The maximal number of rcv that can be associated with the context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_set_dpa_thread
(doca_comch_consumer_completion *consumer_comp,
doca_dpa_thread *dpa_thread)
```

Set the DOCA DPA thread of the completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context.

dpa_thread

The DOCA dpa thread to be associated with the completion context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

This thread will receive notifications for completions.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_set_imm_data_len
(doca_comch_consumer_completion *consumer_comp,
uint32_t imm_data_len)
```

Parameters

consumer_comp

The DOCA Comch consumer completion context to modify.

imm_data_len

Length of immediate data to configure in the consumer completion context.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the length of immediate data supported by a consumer completion context.

The immediate data length of the consumer completion must be greater or equal to that of any associated consumer.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_set_max_num_consumers
(doca_comch_consumer_completion *consumer_comp,
uint32_t max_num_consumers)
```

Set the maximal number of consumers that can be associated with the completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to modify.

max_num_consumers

The maximal number of consumers that can be associated with the context. Will be rounded up to next power of 2

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_completion_set_max_num_recv
(doca_comch_consumer_completion *consumer_comp,
uint32_t max_num_recv)
```

Set the maximal number of receive operations across all consumers associated with the completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to modify.

max_num_recv

The maximal number of recv that can be associated with the context. Will be rounded up to next power of 2

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

`DOCA_EXPERIMENTAL doca_error_t`
`doca_comch_consumer_completion_start`
`(doca_comch_consumer_completion *consumer_comp)`

Start DOCA Comch consumer completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to start.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

On start verifies and finalizes the completion context configuration.

The following is possible for started completion context:

- ▶ Associating DOCA Comch consumers with the completion context.

The following is NOT possible while completion context is started:

- ▶ Setting the properties of the completion context

`DOCA_EXPERIMENTAL doca_error_t`
`doca_comch_consumer_completion_stop`
`(doca_comch_consumer_completion *consumer_comp)`

Stop DOCA Comch consumer completion context.

Parameters

consumer_comp

The DOCA Comch consumer completion context to stop.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

On stop prevents execution of different operations and allows operations that were available before start. For details see [doca_comch_consumer_completion_start\(\)](#). Completion context can't be stopped while there are DOCA Comch consumers associated with it.

The following is possible for stopped completion context:

- ▶ Setting the properties of the completion context

The following is NOT possible while completion context is stopped:

- ▶ Associating DOCA Comch consumers with the completion context.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_create (doca_comch_connection
*comch_connection, doca_mmap *buf_mmap,
doca_comch_consumer **consumer)
```

Parameters

comch_connection

An established control channel connection to create consumer across.

buf_mmap

A registered mmap for the memory region the consumer allows buffer writes to.

consumer

Pointer to pointer to be set to created doca_comch_consumer instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - input parameter is a NULL pointer.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_comch_consumer object or id.
- ▶ DOCA_ERROR_BAD_STATE - comch_connection is not established.
- ▶ DOCA_ERROR_NOT_PERMITTED - incompatible version of comch_connection.

Description

Create a DOCA Comch consumer instance.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_destroy (doca_comch_consumer
*consumer)
```

Parameters

consumer

Pointer to doca_comch_consumer instance to destroy.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - consumer argument is a NULL pointer.
- ▶ DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Destroy a DOCA Comch consumer instance.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_get_dpa_handle
(doca_comch_consumer *consumer,
doca_dpa_dev_comch_consumer_t *dpa_consumer)
```

Retrieve the handle in the dpa memory space of a doca_comch_consumer.

Parameters

consumer

doca_comch_consumer context to get the dpa handle from.

dpa_consumer

A pointer to the handle in the dpa memory space.

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 called before calling doca_ctx_start(), or if not assigned to dpa datapath.

```
DOCA_STABLE doca_error_t doca_comch_consumer_get_id
(const doca_comch_consumer *consumer, uint32_t *id)
```

Parameters

consumer

The doca_comch_consumer instance.

id

Per comch_connection unique id associated with the consumer instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the id the `doca_comch_consumer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_get_imm_data_len (const
doca_comch_consumer *consumer, uint32_t
*imm_data_len)
```

Parameters

consumer

Consumer to query.

imm_data_len

Length of immediate data the consumer is configured to support.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the length of immediate data supported by a consumer.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_set_completion
(doca_comch_consumer *consumer,
doca_comch_consumer_completion *consumer_comp,
uint32_t user_data)
```

Associate consumer with DPA completion context.

Parameters

consumer

The `doca_comch_consumer` instance. Must call `doca_ctx_set_datapath_on_dpa()` prior to this call

consumer_comp

The DOCA Comch consumer completion context to associate with consumer

user_data

User data that can be retrieved in DPA from completion elements returned by this consumer

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

This will allow the completion context to receive completions of this consumer on the DPA

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_consumer_set_dev_max_num_rcv
(doca_comch_consumer *consumer, uint32_t
dev_num_rcv)
```

Set the maximal number of receive operations for a DPA consumer.

Parameters**consumer**

The `doca_comch_consumer` instance.

dev_num_rcv

The maximal number of `rcv` that can be associated with the context. Will be rounded up to next power of 2

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_set_imm_data_len
(doca_comch_consumer *consumer, uint32_t
imm_data_len)
```

Parameters**consumer**

Consumer to set length of.

imm_data_len

Length of immediate data to configure in the consumer.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - cc instance is already active.

Description

Set the length of immediate data supported by a consumer.

Length must be less than or equal to that returned by `doca_comch_consumer_cap_get_max_imm_data_len()`.

```
DOCA_STABLE doca_error_t
doca_comch_consumer_task_post_rcv_alloc_init
(doca_comch_consumer *consumer, doca_buf *buf,
doca_comch_consumer_task_post_rcv **task)
```

Allocate and initialize a `doca_consumer` post receive task.

Parameters

consumer

The `doca_comch_consumer` instance.

buf

Doca buffer available to be populated by producers.

task

Pointer to a `doca_comch_consumer_task_post_rcv` instance populated with input parameters.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NO_MEMORY - no available tasks to allocate.

Description

Doca buffer should be located within the registered mmap associated with consumer instance. Completion callback will be triggered whenever the buffer has been populated by a consumer.

DOCA_STABLE doca_task

```
*doca_comch_consumer_task_post_recv_as_task
(doca_comch_consumer_task_post_recv *task)
```

Parameters

task

The doca_comch_consumer_task_post_recv instance.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert doca_comch_consumer_task_post_recv instance into a generalized task for use with progress engine.

DOCA_STABLE doca_buf

```
*doca_comch_consumer_task_post_recv_get_buf (const
doca_comch_consumer_task_post_recv *task)
```

Parameters

task

The doca_comch_consumer_task_post_recv instance.

Returns

Non NULL upon success, NULL otherwise.

Description

Get the doca_buf from the doca_comch_consumer_task_post_recv instance.

const DOCA_STABLE uint8_t

```
*doca_comch_consumer_task_post_recv_get_imm_data
(const doca_comch_consumer_task_post_recv *task)
```

Parameters

task

The doca_comch_consumer_task_post_recv instance.

Returns

Pointer to immediate data or nullptr if none exists.

Description

Get a pointer to any immediate data from the `doca_comch_consumer_task_post_rcv` instance.

Immediate data will only be set on post rcv completion.

`doca_comch_consumer_task_post_rcv_get_imm_data_len()` indicates the number of valid bits pointed to.

```
DOCA_STABLE uint32_t
doca_comch_consumer_task_post_rcv_get_imm_data_len
(const doca_comch_consumer_task_post_rcv *task)
```

Parameters

task

The `doca_comch_consumer_task_post_rcv` instance.

Returns

The immediate data length on post receive completion.

Description

Get the length of any immediate data from the `doca_comch_consumer_task_post_rcv` instance.

Immediate data length will only be set on post rcv completion. It indicates the valid number of bytes in the pointer return by `doca_comch_consumer_task_post_rcv_get_imm_data()`.

```
DOCA_STABLE uint32_t
doca_comch_consumer_task_post_rcv_get_producer_id
(const doca_comch_consumer_task_post_rcv *task)
```

Parameters

task

The `doca_comch_consumer_task_post_rcv` instance.

Returns

Producer id upon success, 0 otherwise.

Description

Get the producer id from the `doca_comch_consumer_task_post_rcv` instance.

Producer id will only be set on post rcv completion and indicates the remote producer that has written data to the associated `doca_buf`.

DOCA_STABLE void

```
doca_comch_consumer_task_post_rcv_set_buf
(doca_comch_consumer_task_post_rcv *task, doca_buf
*buf)
```

Parameters

task

The `doca_comch_consumer_task_post_rcv` instance.

buf

Buffer to set in the task.

Description

Set the `doca_buf` in a `doca_comch_consumer_task_post_rcv` instance.

DOCA_STABLE doca_error_t

```
doca_comch_consumer_task_post_rcv_set_conf
(doca_comch_consumer *consumer,
doca_comch_consumer_task_post_rcv_completion_cb_t
task_completion_cb,
doca_comch_consumer_task_post_rcv_completion_cb_t
task_error_cb, uint32_t num_post_rcv_tasks)
```

Parameters

consumer

The `doca_comch_consumer` instance.

task_completion_cb

Post receive task completion callback.

task_error_cb

Post receive task error callback.

num_post_rcv_tasks

Number of post_rcv tasks a consumer can allocate. Must not exceed value returned by `doca_comch_consumer_cap_get_max_num_tasks()`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - consumer instance is already active.

Description

Configure the `doca_comch_consumer` post receive task callback and parameters.

2.5.2. DOCA Comch MsgQ

DOCA Comch

DOCA Communication Channel MsgQ library lets you set a direct communication channel between Host/DPU and DPA. The channel is not part of the TCP/IP stack. Please follow the programmer guide for usage instructions.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_consumer_create (doca_comch_msgq
*msgq, doca_comch_consumer **consumer)
```

Create a DOCA Comch consumer instance.

Parameters**msgq**

The DOCA Comch MsgQ to be used for creating the consumer

consumer

Pointer to pointer to be set to created `doca_comch_consumer` instance.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_create (doca_dev *dev,
doca_comch_msgq **msgq)
```

Create a DOCA Comch MsgQ instance.

Parameters

dev

A DOCA device that will be used to send/receive the messages.

msgq

Pointer to pointer to be set to created doca_comch_msgq instance.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_destroy (doca_comch_msgq *msgq)
```

Destroy a DOCA Comch MsgQ instance.

Parameters

msgq

The DOCA Comch MsgQ to destroy.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_producer_create (doca_comch_msgq
*msgq, doca_comch_producer **producer)
```

Create a DOCA Comch producer instance.

Parameters

msgq

The DOCA Comch MsgQ to be used for creating the producer

producer

Pointer to pointer to be set to created doca_comch_producer instance.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_set_dpa_consumer (doca_comch_msgq
*msgq, doca_dpa *dpa)
```

Set consumers to DPA. All consumers on this MsgQ will be created for DPA.

Parameters

msgq

The DOCA Comch MsgQ to modify.

dpa

The DPA context that will use the consumers. Must be on same device as the MsgQ.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

This configuration specifies that all consumers on this MsgQ will be created and used on this DPA instance. By default consumers are set on Host (or DPU).

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_set_dpa_producer (doca_comch_msgq
*msgq, doca_dpa *dpa)
```

Set producers to DPA. All producers on this MsgQ will be created for DPA.

Parameters

msgq

The DOCA Comch MsgQ to modify.

dpa

The DPA context that will use the producers. Must be on same device as the MsgQ.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

This configuration specifies that all producers on this MsgQ will be created and used on this DPA instance. By default producers are set on Host (or DPU).

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_set_max_num_consumers
(doca_comch_msgq *msgq, uint32_t max_num_consumers)
```

Set the maximal number of consumers that can be created using this MsgQ.

Parameters

msgq

The DOCA Comch MsgQ to modify.

max_num_consumers

The maximal number of consumers that can be created.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_set_max_num_producers
(doca_comch_msgq *msgq, uint32_t max_num_producers)
```

Set the maximal number of producers that can be created using this MsgQ.

Parameters

msgq

The DOCA Comch MsgQ to modify.

max_num_producers

The maximal number of producers that can be created.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_msgq_start (doca_comch_msgq *msgq)
```

Start DOCA Comch MsgQ.

Parameters

msgq

The DOCA Comch MsgQ to start.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

On start verifies and finalizes the MsgQ configuration.

The following is possible for started MsgQ:

- ▶ Creating consumers and producers.

The following is NOT possible while completion context is started:

- ▶ Setting the properties of the MsgQ

DOCA_EXPERIMENTAL doca_error_t doca_comch_msgq_stop (doca_comch_msgq *msgq)

Stop DOCA Comch MsgQ.

Parameters

msgq

The DOCA Comch MsgQ to stop.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description

On stop prevents execution of different operations and allows operations that were available before start. For details see [doca_comch_msgq_start\(\)](#). MsgQ can't be stopped while there are DOCA Comch consumers/producers associated with it.

The following is possible for stopped MsgQ:

- ▶ Setting the properties of the MsgQ

The following is NOT possible while MsgQ is stopped:

- ▶ Creating DOCA Comch consumers or producers from MsgQ.

2.5.3. DOCA Comch Producer

DOCA Comch

DOCA Communication Channel Producer offers an extension the `doca_comch` channel for accelerated data transfer between memory on the host and DPU in a FIFO format. An established `doca_comch` connection is required to negotiate the end points of the FIFO. A producer object can populate buffers advertised by any consumers associated with the same

doca_comch connection. The inter-process communication runs over DMA/PCIe and does not affect network bandwidth.

typedef

```
(*doca_comch_producer_task_send_completion_cb_t)
(doca_comch_producer_task_send* task, union doca_data
task_user_data, union doca_data ctx_user_data)
```

Function executed on doca_comch_producer send task completion. Used for both task success and failure.

The implementation can assume this value is not NULL.

typedef uint64_t doca_dpa_dev_comch_producer_t

DPA handle for DPA producer.

```
DOCA_STABLE doca_ctx *doca_comch_producer_as_ctx
(doca_comch_producer *producer)
```

Parameters

producer

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

Returns

Non NULL upon success, NULL otherwise.

Description

Convert doca_comch_producer instance into a generalized context for use with doca core objects.

```
DOCA_STABLE doca_error_t
doca_comch_producer_cap_get_max_buf_list_len (const
doca_devinfo *devinfo, uint32_t *max_buf_list_len)
```

Parameters

devinfo

Devinfo to query the capability for.

max_buf_list_len

Maximum sized buffer list that can be sent by the producer.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - if producer is not supported on device.

Description

Get the max length of `doca_buf` list that can be sent by a `doca_comch_producer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_producer_cap_get_max_buf_size (const
doca_devinfo *devinfo, uint32_t *max_buf_size)
```

Parameters

devinfo

Devinfo to query the capability for.

max_buf_size

Maximum sized buffer that can be sent by the producer.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - if producer is not supported on device.

Description

Get the max size `doca_buf` that can be sent by a `doca_comch_producer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_producer_cap_get_max_num_tasks (const
doca_devinfo *devinfo, uint32_t *max_num_tasks)
```

Parameters

devinfo

Devinfo to query the capability for.

max_num_tasks

The maximum number of tasks that can allocated by the instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the max number of tasks supported by the device for a `doca_comch_producer` instance.

```
DOCA_STABLE doca_error_t
doca_comch_producer_cap_get_max_producers (const
doca_devinfo *devinfo, uint32_t *max_producers)
```

Parameters

devinfo

Devinfo to query the capability for.

max_producers

Maximum number of producers that can be added to a `doca_comch_connection`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the max number of producers that can be associated with a `doca_comch_connection`.

```
DOCA_STABLE doca_error_t
doca_comch_producer_cap_is_supported (const
doca_devinfo *devinfo)
```

Parameters

devinfo

The DOCA device information.

Returns

DOCA_SUCCESS - in case device can implement a producer. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

- ▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo can not implement a producer.

Description

Check if given device is capable of running a producer.

```
DOCA_STABLE doca_error_t doca_comch_producer_create
(doca_comch_connection *comch_connection,
doca_comch_producer **producer)
```

Parameters

comch_connection

An established control channel connection to associate producer with.

producer

Pointer to pointer to be set to created doca_comch_producer instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - input parameter is a NULL pointer.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc doca_comch_producer.
- ▶ DOCA_ERROR_BAD_STATE - comch_connection is not established.
- ▶ DOCA_ERROR_NOT_PERMITTED - incompatible version of comch_connection.

Description

Create a DOCA Comch producer instance.

```
DOCA_STABLE doca_error_t
doca_comch_producer_destroy (doca_comch_producer
*producer)
```

Parameters

producer

Pointer to doca_comch_producer instance to destroy.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - producer argument is a NULL pointer.

- ▶ DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Destroy a DOCA Comch producer instance.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_producer_dpa_completion_attach
(doca_comch_producer *producer, doca_dpa_completion
*dpa_comp)
```

Associate producer with DPA completion context.

Parameters

producer

The doca_comch_producer instance. Must call [doca_ctx_set_datapath_on_dpa\(\)](#) prior to this call

dpa_comp

The DOCA DPA completion context to associate with producer.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

Description

This will allow the completion context to send completions of this producer on the DPA

```
DOCA_EXPERIMENTAL doca_error_t
doca_comch_producer_get_dpa_handle
(doca_comch_producer *producer,
doca_dpa_dev_comch_producer_t *dpa_producer)
```

Retrieve the handle in the dpa memory space of a doca_comch_producer.

Parameters

producer

doca_comch_producer context to get the dpa handle from.

dpa_producer

A pointer to the handle in the dpa memory space.

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 called before calling `doca_ctx_start()`, or if not assigned to dpa datapath.

DOCA_STABLE `doca_error_t doca_comch_producer_get_id`
 (`const doca_comch_producer *producer, uint32_t *id`)

Parameters

producer

The `doca_comch_producer` instance.

id

Per `comch_connection` unique id associated with the producer instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description

Get the id the `doca_comch_producer` instance.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_comch_producer_set_dev_max_num_send`
 (`doca_comch_producer *producer, uint32_t dev_num_send`)

Set the maximal number of send operations for a DPA producer.

Parameters

producer

The `doca_comch_producer` instance.

dev_num_send

The maximal number of send that can be associated with the context. Will be rounded up to next power of 2

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_STABLE doca_error_t
doca_comch_producer_task_send_alloc_init
(doca_comch_producer *producer, const doca_buf *buf,
uint8_t *imm_data, uint32_t imm_data_len, uint32_t
consumer_id, doca_comch_producer_task_send **task)
```

Parameters

producer

The doca_comch_producer instance.

buf

Doca buffer to send to a consumer.

imm_data

Pointer to immediate data to include in the send.

imm_data_len

Length of data in bytes pointed to by imm_data.

consumer_id

ID of consumer to send the buffer to.

task

Pointer to a doca_comch_producer_task_send instance populated with input parameters.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_NO_MEMORY - no available tasks to allocate.

Description

Allocate and initialize a doca_producer send task.

```
DOCA_STABLE doca_task
*doca_comch_producer_task_send_as_task
(doca_comch_producer_task_send *task)
```

Parameters

task

The doca_comch_producer_task_send instance.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert `doca_comch_producer_task_send` instance into a generalized task for use with progress engine.

DOCA_STABLE `doca_buf`

```
*doca_comch_producer_task_send_get_buf (const  
doca_comch_producer_task_send *task)
```

Parameters

task

The `doca_comch_producer_task_send` instance.

Returns

Non NULL upon success, NULL otherwise.

Description

Get the `doca_buf` from the `doca_comch_producer_task_send` instance.

DOCA_STABLE `uint32_t`

```
doca_comch_producer_task_send_get_consumer_id (const  
doca_comch_producer_task_send *task)
```

Parameters

task

The `doca_comch_producer_task_send` instance.

Returns

`Consumer_id`

Description

Get the consumer id from the `doca_comch_producer_task_send` instance.

```
DOCA_STABLE uint8_t
doca_comch_producer_task_send_get_imm_data (const
doca_comch_producer_task_send *task)
```

Parameters

task

The doca_comch_producer_task_send instance.

Returns

Immediate data pointer or nullptr if none exists

Description

Get any immediate data from the doca_comch_producer_task_send instance.

```
DOCA_STABLE uint32_t
doca_comch_producer_task_send_get_imm_data_len
(const doca_comch_producer_task_send *task)
```

Parameters

task

The doca_comch_producer_task_send instance.

Returns

Length of immediate data.

Description

Get any immediate data length from the doca_comch_producer_task_send instance.

```
DOCA_STABLE void
doca_comch_producer_task_send_set_buf
(doca_comch_producer_task_send *task, const doca_buf
*buf)
```

Parameters

task

The doca_comch_producer_task_send instance.

buf

Buffer to set in the task.

Description

Set the `doca_buf` in a `doca_comch_producer_task_send` instance.

```
DOCA_STABLE doca_error_t
doca_comch_producer_task_send_set_conf
(doca_comch_producer *producer,
doca_comch_producer_task_send_completion_cb_t
task_completion_cb,
doca_comch_producer_task_send_completion_cb_t
task_error_cb, uint32_t num_send_tasks)
```

Parameters**producer**

The `doca_comch_producer` instance.

task_completion_cb

Send task completion callback.

task_error_cb

Send task error callback.

num_send_tasks

Number of send tasks a producer can allocate. Must not exceed value returned by `doca_comch_producer_cap_get_max_num_tasks()`

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- ▶ DOCA_ERROR_BAD_STATE - producer instance is already active.

Description

Configure the `doca_comch_producer` send task callback and parameters.

```
DOCA_STABLE void
doca_comch_producer_task_send_set_consumer_id
(doca_comch_producer_task_send *task, const uint32_t
consumer_id)
```

Parameters

task

The `doca_comch_producer_task_send` instance.

consumer_id

Consumer id to set in the task.

Description

Set the consumer id in the `doca_comch_producer_task_send` instance.

DOCA_STABLE void

```
doca_comch_producer_task_send_set_imm_data
(doca_comch_producer_task_send *task, uint8_t
*imm_data, uint32_t imm_data_len)
```

Parameters

task

The `doca_comch_producer_task_send` instance.

imm_data

Pointer to immediate data to copy to task

imm_data_len

Length of the immediate data in bytes

Description

Set immediate data in the `doca_comch_producer_task_send` instance.

2.6. DOCA 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 doca_event_handle_t doca_event_channel_t
```

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

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_NO_MEMORY if a memory related error has occurred. DOCA_ERROR_NOT_CONNECTED if there is no connection. DOCA_ERROR_UNKNOWN if an unknown error occurred.

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL `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_EXPERIMENTAL `doca_error_t`
`doca_comm_channel_ep_get_peer_addr_list`
`(const doca_comm_channel_ep_t *local_ep,`
`doca_comm_channel_addr_tpeer_addr_array,`
`uint32_t *peer_addr_array_len)`

get an array of the peer_addr connected to a given service object

Parameters

local_ep

Pointer to service endpoint to get peer_addr array for.

peer_addr_array

An array of connected peer_addr objects.

peer_addr_array_len

The number of entries in the output peer_addr_array.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL. DOCA_ERROR_NOT_SUPPORTED if called on a client endpoint.

Description

This function can only be called on the service side. This function will return an array of all peer_addr connected to the given service endpoint, based on the information that was updated at the last time [doca_comm_channel_ep_update_service_state_info\(\)](#) was called.

**Note:**

When calling [doca_comm_channel_ep_update_service_state_info\(\)](#) any previously received peer_addr_array is invalidated.

DOCA_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_get_pending_connections
 (const doca_comm_channel_ep_t *local_ep, uint32_t
 *pending_connections)

get the number of pending connections for a given service endpoint

Parameters**local_ep**

Pointer to peer_addr to get pending connections for.

pending_connections

The number of pending connections for the given service endpoint.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL. DOCA_ERROR_NOT_SUPPORTED if called on a client endpoint.

Description

This function can only be called on the service side. This function will return the number of pending connections for the given service endpoint, based on the information that was updated at the last time [doca_comm_channel_ep_update_service_state_info\(\)](#) was called.

Pending connections are connections that are waiting for handshake to be completed. `doca_comm_channel_ep_rcvfrom()` should be called to handle pending connections.

```
DOCA_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_get_rcv_queue_size
(doca_comm_channel_ep_t *ep, uint16_t
*rcv_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_rcv_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_rcv_queue_size()`, a default value is used and can be inquired by calling `doca_comm_channel_ep_get_rcv_queue_size()`.

Parameters

ep

endpoint from which the property should be retrieved.

rcv_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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_get_service_event_channel
(doca_comm_channel_ep_t *local_ep,
 doca_event_channel_t *service_event_channel)
```

Extract the service_event_channel handle for user's use This handle can be used to get interrupt when one of the following events occurred: new client connected, client disconnected or service moved to error state using epoll() or similar function. If an event was received, the application can call doca_comm_channel_ep_update_service_state_info() and it's query functions to get the current service state. The service event channel is armed automatically upon calling doca_comm_channel_ep_update_service_state_info().

Parameters

local_ep

handle for the endpoint created beforehand with [doca_comm_channel_ep_create\(\)](#).

service_event_channel

handle for service event channel.

Returns

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep was provided or if service_event_channel is NULL. DOCA_ERROR_BAD_STATE if called before calling [doca_comm_channel_ep_listen\(\)](#). DOCA_ERROR_NOT_SUPPORTED if called on a non-service instant.

Description

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\(\)](#).

This function available only for service side use.

```
DOCA_EXPERIMENTAL 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 defined 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_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_rcvfrom
(doca_comm_channel_ep_t *local_ep, void *msg,
size_t *len, int flags, doca_comm_channel_addr_t
**peer_addr)
```

Receive message from connected client/service.

Parameters

local_ep

handle for the endpoint created beforehand with [doca_comm_channel_ep_create\(\)](#).

msg

pointer to the buffer where the message should be stored.

len

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

flags

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_rcvfrom\(\)](#) also used for accepting new connection from clients.

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_set_rcv_queue_size
(doca_comm_channel_ep_t *ep, uint16_t
rcv_queue_size)
```

set receive queue size property for endpoint. The value `rcv_queue_size` may be increased internally, the actual value can be queried using `doca_comm_channel_ep_get_rcv_queue_size()`.

Parameters

ep

endpoint to set the property for.

rcv_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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_ep_update_service_state_info
(doca_comm_channel_ep_t *local_ep)
```

update the connections status for a given service endpoint

Parameters

local_ep

Pointer to endpoint to update the connections status on.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if local_ep is NULL. DOCA_ERROR_NOT_SUPPORTED if called on a client endpoint. DOCA_ERROR_DRIVER if failed to query the service status. DOCA_ERROR_AGAIN if an unexpected number of new clients joined and service status needs to be queried again. DOCA_ERROR_CONNECTION_RESET if the the service is in error state.

Description

Can only be called on the service side. This function saves a snapshot of the current service state, which can be queried using the functions [doca_comm_channel_ep_get_peer_addr_list\(\)](#) or [doca_comm_channel_ep_get_pending_connections\(\)](#). This function can also be used to check if service is in error state, in that case it cannot be recovered and needs to be destroyed.



Note:

Calling this function will also invalidate any peer_addr_array received from previous calls to [doca_comm_channel_ep_get_peer_addr_list\(\)](#).

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_get_max_rcv_queue_size
(doca_devinfo *devinfo, uint32_t
*max_rcv_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_rcv_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_rcv_queue_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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 ('\0').

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if max_service_name_len is NULL.

```
DOCA_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_peer_addr_get_rcv_messages
(const doca_comm_channel_addr_t *peer_addr,
uint64_t *rcv_messages)
```

get total messages received from specific peer address

Parameters

peer_addr

Pointer to peer_addr to query statistics for.

rcv_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_EXPERIMENTAL 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_EXPERIMENTAL doca_error_t
doca_comm_channel_peer_addr_get_send_in_flight_message
(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_EXPERIMENTAL 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_EXPERIMENTAL 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_EXPERIMENTAL 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_rcvfrom\(\)](#). The user_context for new peers is initialized to 0.

```
DOCA_EXPERIMENTAL 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.

2.7. DOCA 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"

To remove warnings of deprecated compile with "-D DOCA_ALLOW_DEPRECATED_API"

#define __attribute__

To allow compiling functions and structs that are using GCC attributes using __attribute__() in compilers that don't support them.

#define DLL_EXPORT_ATTR dllimport

To hint the compiler that the function should be exposed to the DLL or imported from it, according to the availability of DOCA_EXPORTS. By default, it should be imported.

#define DOCA_DEPRECATED

```
__declspec(deprecated("Symbol is defined as deprecated")), DLL_EXPORT_ATTR)
```

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

```
#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_STABLE
__declspec(DLL_EXPORT_ATTR)
```

To set a Symbol (or specifically a function) as stable API, i.e. it won't be changed.

```
#define DOCA_STRUCT_START size_t
__doca_api_version
```

Compatibility Helpers

2.8. DOCA Compress Engine

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

```
typedef
(*doca_compress_task_compress_deflate_completion_cb_t)
(doca_compress_task_compress_deflate* task,
union doca_data task_user_data, union doca_data
ctx_user_data)
```

Function to execute on compress deflate task completion.

```
typedef
(*doca_compress_task_decompress_deflate_completion_cb_t)
(doca_compress_task_decompress_deflate* task,
union doca_data task_user_data, union doca_data
ctx_user_data)
```

Function to execute on decompress deflate task completion.

```
typedef
(*doca_compress_task_decompress_lz4_block_completion_cb)
(doca_compress_task_decompress_lz4_block* task,
union doca_data task_user_data, union doca_data
ctx_user_data)
```

Function to execute on decompress LZ4 block task completion.

```
typedef
(*doca_compress_task_decompress_lz4_stream_completion_cb)
(doca_compress_task_decompress_lz4_stream*
task, union doca_data task_user_data, union
doca_data ctx_user_data)
```

Function to execute on decompress LZ4 stream task completion.

```
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

Adapt doca_compress instance into a generalized context for use with doca core objects.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_get_max_num_tasks
(doca_compress *compress, uint32_t
*max_num_tasks)
```

Parameters

compress

Compress context to get max number of tasks from

max_num_tasks

Sum of num tasks should not exceed this number (

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 number of tasks

This method retrieves the maximum number of tasks for a device. Sum of num tasks should not exceed this number.

See also:

[doca_compress_task_compress_deflate_set_conf](#),
[doca_compress_task_decompress_deflate_set_conf](#),
[doca_compress_task_decompress_lz4_set_conf](#))

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_compress_deflate_get_max_buf_lis
(const doca_devinfo *devinfo, uint32_t
*max_buf_list_len)
```

Parameters

devinfo

The DOCA device information.

max_buf_list_len

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 deflate task.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_compress_deflate_get_max_buf_size(
    const doca_devinfo *devinfo, uint64_t
    *max_buffer_size)
```

Get compress deflate max size.

Parameters**devinfo**

doca device info to check

max_buffer_size

The max buffer size for compress deflate operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method retrieves a compress deflate max size for a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_compress_deflate_is_supported
(const doca_devinfo *devinfo)
```

Check if a compress deflate task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - failed to query the device for its capabilities.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method checks if a compress deflate task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_deflate_get_max_buf
(const doca_devinfo *devinfo, uint32_t
*max_buf_list_len)
```

Parameters

devinfo

The DOCA device information.

max_buf_list_len

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 decompress deflate task.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_deflate_get_max_buf
(const doca_devinfo *devinfo, uint64_t
*max_buffer_size)
```

Get decompress deflate max size.

Parameters

devinfo

doca device info to check

max_buffer_size

The max buffer size for decompress deflate operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method retrieves decompress deflate max size

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_deflate_is_supported
(const doca_devinfo *devinfo)
```

Check if a decompress deflate task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - failed to query the device for its capabilities.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method checks if a decompress deflate task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_block_get_max_b
(const doca_devinfo *devinfo, uint32_t
*max_buf_list_len)
```

Parameters

devinfo

The DOCA device information.

max_buf_list_len

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 decompress LZ4 block task.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_block_get_max_b
(const doca_devinfo *devinfo, uint64_t
*max_buffer_size)
```

Get decompress LZ4 block max size.

Parameters

devinfo

doca device info to check

max_buffer_size

The max buffer size for decompress LZ4 block operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method retrieves decompress LZ4 block max size

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_block_is_supporte
(const doca_devinfo *devinfo)
```

Check if a decompress LZ4 block task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - failed to query the device for its capabilities.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method checks if a decompress LZ4 block task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_stream_get_max
(const doca_devinfo *devinfo, uint32_t
*max_buf_list_len)
```

Parameters

devinfo

The DOCA device information.

max_buf_list_len

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 decompress LZ4 stream task

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_stream_get_max
(const doca_devinfo *devinfo, uint64_t
*max_buffer_size)
```

Get decompress LZ4 stream max size.

Parameters

devinfo

doca device info to check

max_buffer_size

The max buffer size for decompress LZ4 stream operation in bytes.

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method retrieves decompress LZ4 stream max size

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_cap_task_decompress_lz4_stream_is_supported
(const doca_devinfo *devinfo)
```

Check if a decompress LZ4 stream task is supported by a device.

Parameters

devinfo

doca device info to check

Returns

DOCA_SUCCESS - in case device supports the task Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - failed to query the device for its capabilities.
- ▶ DOCA_ERROR_NOT_SUPPORTED - devinfo does not support the task.

Description

This method checks if a decompress LZ4 stream task is supported by a device

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_create (doca_dev *dev,
doca_compress **compress)
```

Parameters

dev

The device to attach to the compress context

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 - one or more of the arguments is null.
- ▶ DOCA_ERROR_NO_MEMORY - failed to alloc `doca_compress`.
- ▶ DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.

Description

Create a DOCA COMPRESS instance.

```
DOCA_EXPERIMENTAL 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

Description

Destroy a DOCA COMPRESS instance.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_compress_deflate_alloc_init
(doca_compress *compress, const doca_buf
*src_buff, doca_buf *dst_buff, doca_data user_data,
doca_compress_task_compress_deflate **task)
```

Allocate `compress` deflate task.

Parameters

compress

The `compress` context to allocate the task from

src_buff

Source buffer

dst_buff

Destination buffer

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

Returns

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

- ▶ DOCA_ERROR_NO_MEMORY - all compress deflate tasks are already allocated.
- ▶ DOCA_ERROR_INVALID_VALUE - can not initialize source HW resources.

Description

This method allocates and initializes a compress deflate task. Task parameters can be set later on by setters.

DOCA_EXPERIMENTAL doca_task

```
*doca_compress_task_compress_deflate_as_task  
(doca_compress_task_compress_deflate *task)
```

convert compress deflate task to doca_task

Parameters**task**

The task to convert

Returns

doca_task

DOCA_EXPERIMENTAL uint32_t

```
doca_compress_task_compress_deflate_get_adler_cs  
(const doca_compress_task_compress_deflate *task)
```

get compress deflate task adler checksum

Parameters**task**

Task to get the destination from

Returns

adler

Description

**Note:**

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL uint32_t
doca_compress_task_compress_deflate_get_crc_cs
(const doca_compress_task_compress_deflate *task)
get compress deflate task CRC checksum
```

Parameters

task

Task to get the destination from

Returns

CRC

Description

**Note:**

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL doca_buf
*doca_compress_task_compress_deflate_get_dst
(const doca_compress_task_compress_deflate *task)
get compress deflate task destination
```

Parameters

task

Task to get the destination from

Returns

destination buffer

```

const DOCA_EXPERIMENTAL doca_buf
*doca_compress_task_compress_deflate_get_src
(const doca_compress_task_compress_deflate *task)

```

get compress deflate task source

Parameters

task

Task to get the source from

Returns

source buffer

```

DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_compress_deflate_set_conf
(doca_compress *compress,
doca_compress_task_compress_deflate_completion_cb_t
task_completion_cb,
doca_compress_task_compress_deflate_completion_cb_t
task_error_cb, uint32_t num_tasks)

```

This method sets the compress deflate task configuration.

Parameters

compress

The compress context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of compress deflate tasks that the context can allocate

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

```
DOCA_EXPERIMENTAL void
doca_compress_task_compress_deflate_set_dst
(doca_compress_task_compress_deflate *task,
doca_buf *dst_buff)
```

set compress deflate task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

```
DOCA_EXPERIMENTAL void
doca_compress_task_compress_deflate_set_src
(doca_compress_task_compress_deflate *task, const
doca_buf *src_buff)
```

set compress deflate task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_deflate_alloc_init
(doca_compress *compress, const doca_buf
*src_buff, doca_buf *dst_buff, doca_data user_data,
doca_compress_task_decompress_deflate **task)
```

Allocate decompress deflate task.

Parameters

compress

The compress context to allocate the task from

src_buff

Source buffer

dst_buff

Destination buffer

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

Returns

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

- ▶ DOCA_ERROR_NO_MEMORY - all compress deflate tasks are already allocated.

Description

This method allocates and initializes a decompress deflate task. Task parameters can be set later on by setters.

DOCA_EXPERIMENTAL doca_task

***doca_compress_task_decompress_deflate_as_task
(doca_compress_task_decompress_deflate *task)**

convert decompress deflate task to doca_task

Parameters**task**

The task to convert

Returns

doca_task

```
DOCA_EXPERIMENTAL uint32_t
doca_compress_task_decompress_deflate_get_adler_cs
(const doca_compress_task_decompress_deflate
*task)
```

get decompress deflate task Adler checksum

Parameters

task

Task to get the destination from

Returns

Adler

Description



Note:

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL uint32_t
doca_compress_task_decompress_deflate_get_crc_cs
(const doca_compress_task_decompress_deflate
*task)
```

get decompress deflate task CRC checksum

Parameters

task

Task to get the destination from

Returns

CRC

Description



Note:

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL doca_buf  
*doca_compress_task_decompress_deflate_get_dst  
(const doca_compress_task_decompress_deflate  
*task)
```

get decompress deflate task destination

Parameters

task

Task to get the destination from

Returns

destination buffer

```
const DOCA_EXPERIMENTAL doca_buf  
*doca_compress_task_decompress_deflate_get_src  
(const doca_compress_task_decompress_deflate  
*task)
```

get decompress deflate task source

Parameters

task

Task to get the source from

Returns

source buffer

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_deflate_set_conf
(doca_compress *compress,
doca_compress_task_decompress_deflate_completion_cb_t
task_completion_cb,
doca_compress_task_decompress_deflate_completion_cb_t
task_error_cb, uint32_t num_tasks)
```

This method sets the decompress deflate task configuration.

Parameters

compress

The compress context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of decompress deflate tasks that the context can allocate

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_INVALID_VALUE - can not initialize source HW resources.

DOCA_EXPERIMENTAL void

```
doca_compress_task_decompress_deflate_set_dst
(doca_compress_task_decompress_deflate *task,
doca_buf *dst_buff)
```

set decompress deflate task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

```
DOCA_EXPERIMENTAL void
doca_compress_task_decompress_deflate_set_src
(doca_compress_task_decompress_deflate *task,
const doca_buf *src_buff)
```

set decompress deflate task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_lz4_block_alloc_init
(doca_compress *compress, const doca_buf
*src_buff, doca_buf *dst_buff, doca_data user_data,
doca_compress_task_decompress_lz4_block **task)
```

Allocate decompress LZ4 block task.

Parameters

compress

The compress context to allocate the task from

src_buff

Source buffer

dst_buff

Destination buffer

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

Returns

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

- ▶ DOCA_ERROR_NO_MEMORY - all decompress LZ4 block tasks are already allocated.
- ▶ DOCA_ERROR_INVALID_VALUE - can not initialize source HW resources.

Description

This method allocates and initializes a decompress LZ4 block task. Task parameters can be set later on by setters.

DOCA_EXPERIMENTAL doca_task

```
*doca_compress_task_decompress_lz4_block_as_task
(doca_compress_task_decompress_lz4_block *task)
```

convert decompress LZ4 block task to doca_task

Parameters

task

The task to convert

Returns

doca_task

DOCA_EXPERIMENTAL uint32_t

```
doca_compress_task_decompress_lz4_block_get_crc_cs
(const doca_compress_task_decompress_lz4_block
*task)
```

get decompress LZ4 block task CRC checksum

Parameters

task

Task to get the destination from

Returns

CRC checksum of the block

Description



Note:

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL doca_buf  
*doca_compress_task_decompress_lz4_block_get_dst  
(const doca_compress_task_decompress_lz4_block  
*task)
```

get decompress LZ4 block task destination

Parameters

task

Task to get the destination from

Returns

destination buffer

```
const DOCA_EXPERIMENTAL doca_buf  
*doca_compress_task_decompress_lz4_block_get_src  
(const doca_compress_task_decompress_lz4_block  
*task)
```

get decompress LZ4 block task source

Parameters

task

Task to get the source from

Returns

source buffer

```
DOCA_EXPERIMENTAL uint32_t  
doca_compress_task_decompress_lz4_block_get_xxh_cs  
(const doca_compress_task_decompress_lz4_block  
*task)
```

get decompress LZ4 block task xxHash-32 checksum

Parameters

task

Task to get the checksums from

Returns

xxHash-32 checksum of the block

Description



Note:

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_lz4_block_set_conf
(doca_compress *compress,
doca_compress_task_decompress_lz4_block_completion_cb_t
task_completion_cb,
doca_compress_task_decompress_lz4_block_completion_cb_t
task_error_cb, uint32_t num_tasks)
```

This method sets the decompress LZ4 block task configuration.

Parameters

compress

The compress context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of decompress LZ4 block tasks that the context can allocate

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

```
DOCA_EXPERIMENTAL void  
doca_compress_task_decompress_lz4_block_set_dst  
(doca_compress_task_decompress_lz4_block *task,  
doca_buf *dst_buff)
```

set decompress LZ4 block task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

```
DOCA_EXPERIMENTAL void  
doca_compress_task_decompress_lz4_block_set_src  
(doca_compress_task_decompress_lz4_block *task,  
const doca_buf *src_buff)
```

set decompress LZ4 block task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_lz4_stream_alloc_init
(doca_compress *compress, uint8_t
has_block_checksum, uint8_t
are_blocks_independent, const doca_buf *src_buff,
doca_buf *dst_buff, doca_data user_data,
doca_compress_task_decompress_lz4_stream
**task)
```

Allocate decompress LZ4 stream task.

Parameters

compress

The compress context to allocate the task from

has_block_checksum

1 if the task should expect blocks in the stream to have a checksum, 0 otherwise

are_blocks_independent

1 the the task should expect blocks to be independent, 0 otherwise (dependent blocks)

src_buff

Source buffer

dst_buff

Destination buffer

user_data

doca_data that can be retrieved from the task (usually when the task is completed).

task

The allocated task

Returns

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

- ▶ DOCA_ERROR_NO_MEMORY - all decompress LZ4 stream tasks are already allocated.
- ▶ DOCA_ERROR_INVALID_VALUE - can not initialize source HW resources.

Description

This method allocates and initializes a decompress LZ4 stream task. Task parameters can be set later on by setters.

```
DOCA_EXPERIMENTAL doca_task
*doca_compress_task_decompress_lz4_stream_as_task
(doca_compress_task_decompress_lz4_stream
*task)
```

convert decompress LZ4 stream task to doca_task

Parameters

task

The task to convert

Returns

doca_task

```
DOCA_EXPERIMENTAL uint8_t
doca_compress_task_decompress_lz4_stream_get_are_block
(const doca_compress_task_decompress_lz4_stream
*task)
```

get decompress LZ4 stream task are_blocks_independent flag

Parameters

task

Task to get the are_blocks_independent flag from

Returns

1 if the task should expect blocks to be independent, 0 otherwise (dependent blocks)

```
DOCA_EXPERIMENTAL uint32_t
doca_compress_task_decompress_lz4_stream_get_crc_cs
(const doca_compress_task_decompress_lz4_stream
*task)
```

get decompress LZ4 stream task CRC checksum

Parameters

task

Task to get the destination from

Returns

CRC checksum of the stream

Description

**Note:**

Valid only on successful completion of the task. Otherwise, undefined behavior.

DOCA_EXPERIMENTAL doca_buf

```
*doca_compress_task_decompress_lz4_stream_get_dst  
(const doca_compress_task_decompress_lz4_stream  
*task)
```

get decompress LZ4 stream task destination

Parameters

task

Task to get the destination from

Returns

destination buffer

DOCA_EXPERIMENTAL uint8_t

```
doca_compress_task_decompress_lz4_stream_get_has_block  
(const doca_compress_task_decompress_lz4_stream  
*task)
```

get decompress LZ4 stream task has_block_checksum flag

Parameters

task

Task to get the has_block_checksum flag from

Returns

1 if the task should expect blocks in the stream to have a checksum, 0 otherwise

```
const DOCA_EXPERIMENTAL doca_buf
*doca_compress_task_decompress_lz4_stream_get_src
(const doca_compress_task_decompress_lz4_stream
*task)
```

get decompress LZ4 stream task source

Parameters

task

Task to get the source from

Returns

source buffer

```
DOCA_EXPERIMENTAL uint32_t
doca_compress_task_decompress_lz4_stream_get_xxh_cs
(const doca_compress_task_decompress_lz4_stream
*task)
```

get decompress LZ4 stream task xxHash-32 checksum

Parameters

task

Task to get the checksums from

Returns

xxHash-32 checksum of the stream

Description



Note:

Valid only on successful completion of the task. Otherwise, undefined behavior.

```
DOCA_EXPERIMENTAL void
doca_compress_task_decompress_lz4_stream_set_are_block
(doca_compress_task_decompress_lz4_stream
*task, uint8_t are_blocks_independent)
```

set decompress LZ4 stream task are_blocks_independent flag

Parameters

task

Task to set the source to

are_blocks_independent

1 the the task should expect blocks to be independent, 0 otherwise (dependent blocks)

```
DOCA_EXPERIMENTAL doca_error_t
doca_compress_task_decompress_lz4_stream_set_conf
(doca_compress *compress,
doca_compress_task_decompress_lz4_stream_completion_cb
task_completion_cb,
doca_compress_task_decompress_lz4_stream_completion_cb
task_error_cb, uint32_t num_tasks)
```

This method sets the decompress LZ4 stream task configuration.

Parameters

compress

The compress context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_tasks

Number of decompress LZ4 stream tasks that the context can allocate

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

DOCA_EXPERIMENTAL void
doca_compress_task_decompress_lz4_stream_set_dst
 (doca_compress_task_decompress_lz4_stream
 *task, doca_buf *dst_buff)

set decompress LZ4 stream task destination

Parameters

task

Task to set the destination to

dst_buff

destination buffer to set

DOCA_EXPERIMENTAL void
doca_compress_task_decompress_lz4_stream_set_has_block
 (doca_compress_task_decompress_lz4_stream
 *task, uint8_t has_block_checksum)

set decompress LZ4 stream task has_block_checksum flag

Parameters

task

Task to set the source to

has_block_checksum

1 if the task should expect blocks in the stream to have a checksum, 0 otherwise

DOCA_EXPERIMENTAL void
doca_compress_task_decompress_lz4_stream_set_src
 (doca_compress_task_decompress_lz4_stream
 *task, const doca_buf *src_buff)

set decompress LZ4 stream task source

Parameters

task

Task to set the source to

src_buff

Source buffer to set

2.9. DOCA Environment Configurations

#define DOCA_COMPAT_HELPERS

declares the support/need for compatibility helper utils

2.10. DOCA Device Emulation

DOCA Device Emulation - PCI Devices

DOCA Device Emulation - Virtio FS Devices

DOCA Device Emulation - Virtio Devices

2.10.1. DOCA Device Emulation - PCI Devices

DOCA Device Emulation

DOCA library for emulated PCI devices

DOCA Device Emulation - PCI Device Types

enum doca_devemu_pci_hotplug_state

DOCA devemu pci hotplug state.

The steps for hotplug a device are: 1. check the current hotplug state

1.1 if state == DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF then issue hotplug operation (call [doca_devemu_pci_dev_hotplug\(\)](#)) and wait for transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_ON. 1.2 if state == DOCA_DEVEMU_PCI_HP_STATE_PLUG_IN_PROGRESS then wait for transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_ON. 1.3 if state == DOCA_DEVEMU_PCI_HP_STATE_UNPLUG_IN_PROGRESS then wait for transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF and go to step 1.1. 1.4 if state == DOCA_DEVEMU_PCI_HP_STATE_POWER_ON then do nothing --> device is plugged.

The steps for hot unplug a device are: 1. check the current hotplug state

1.1 if state == DOCA_DEVEMU_PCI_HP_STATE_POWER_ON then issue hot unplug operation (call [doca_devemu_pci_dev_hotunplug\(\)](#)) and wait for transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF. 1.2 if state == DOCA_DEVEMU_PCI_HP_STATE_UNPLUG_IN_PROGRESS then wait for

transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF. 1.3 if state == DOCA_DEVEMU_PCI_HP_STATE_PLUG_IN_PROGRESS then wait for transition to DOCA_DEVEMU_PCI_HP_STATE_POWER_ON and go to step 1.1 or issue hot unplug operation (call `doca_devemu_pci_dev_hotunplug()`) then wait for DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF. 1.4 if state == DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF then do nothing --> device is un-plugged.

**Note:**

It is recommended to use `doca_devemu_pci_dev_event_hotplug_state_change` mechanism to get notifications on hotplug state changes.

Values

DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF = 0

Device is powered off and not visible by the host. Device is in transitional state to become un-plugged from host.

DOCA_DEVEMU_PCI_HP_STATE_UNPLUG_IN_PROGRESS

Device is in transitional state to become plugged to host.

DOCA_DEVEMU_PCI_HP_STATE_PLUG_IN_PROGRESS

Device is powered on and visible by the host.

DOCA_DEVEMU_PCI_HP_STATE_POWER_ON

typedef

```
(*doca_devemu_pci_dev_event_bar_stateful_region_driver_write_handler)
(doca_devemu_pci_dev_event_bar_stateful_region_driver_write*
event, union doca_data user_data)
```

Function to be executed on PCI write transactions to BAR stateful region.

```
typedef (*doca_devemu_pci_dev_event_flr_handler_cb_t)
(doca_devemu_pci_dev* pci_dev, union doca_data
user_data)
```

Function to be executed on PCI FLR (Function Level Reset). The event handler will enable users to quiesce, flush and reset the necessary resources associated with the emulated PCI device. Upon event, all PCI I/O transactions to/from the host memory are disabled. Additionally, the user should re-configure the emulated PCI device. This re-configuration requires flushing of all the outstanding resources associated with the emulated PCI device, which were initially owned by the PCI device and moved the the ownership of the user. The re-configuration also requires destruction of all the associated resources (e.g. DBs, MSIXs, MMAPs), resetting the associated emulated PCI device (perform `stop()` and `start()` operations) and re-creating all the needed resources.

typedef

```
(*doca_devemu_pci_dev_event_hotplug_state_change_handler_cb_t)
(doca_devemu_pci_dev* pci_dev, union doca_data
user_data)
```

Function to be executed on hotplug state change event occurrence.

typedef uint64_t

```
doca_dpa_dev_devemu_pci_db_completion_t
```

DPA handle for emulated PCI device doorbell completion context.

```
typedef uint64_t doca_dpa_dev_devemu_pci_db_t
```

DPA handle for emulated PCI device doorbell.

```
typedef uint64_t doca_dpa_dev_devemu_pci_msix_t
```

DPA handle for emulated PCI device MSI-X.

```
DOCA_EXPERIMENTAL doca_error_t
```

```
doca_devemu_pci_cap_get_max_hotplug_devices (const
doca_devinfo *devinfo, uint32_t *max_hotplug_devices)
```

Get the maximum number of PCI devices, across all PCI types, that can be hot-plugged by the device.

Parameters

devinfo

The device to query.

max_hotplug_devices

Number of PCI devices that can be hot plugged by the device.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - internal doca driver error.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_create (doca_dpa_thread
*th, doca_devemu_pci_db_completion **db_comp)
```

Allocate DOCA devemu PCI device doorbell completion context on DPA. The created completion context will be associated with a single dpa thread.

Parameters

th

The DOCA dpa thread to be associated with the completion context.

db_comp

The newly created DOCA devemu PCI device doorbell completion context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_destroy
(doca_devemu_pci_db_completion *db_comp)
```

Destroy the DOCA devemu PCI device doorbell completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to destroy. Must be stopped.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

The associated dpa handle will be destroyed as well.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_get_curr_num_dbs
(doca_devemu_pci_db_completion *db_comp, uint32_t
*num_dbs)
```

Get the current number of doorbells that are associated with the completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to query.

num_dbs

The current number of doorbells that are associated with the context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_get_dpa_handle
(doca_devemu_pci_db_completion *db_comp,
doca_dpa_dev_devemu_pci_db_completion_t
*db_comp_handle)
```

Get the DPA handle for the DOCA devemu PCI device doorbell completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context previously created on DPA.

db_comp_handle

A pointer to the associated DPA handle in the dpa memory space.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_get_max_num_dbs
(doca_devemu_pci_db_completion *db_comp, uint32_t
*num_dbs)
```

Get the maximal number of doorbells that can be associated with the completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to query.

num_dbs

The maximal number of doorbells that can be associated with the context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_set_max_num_dbs
(doca_devemu_pci_db_completion *db_comp, uint32_t
num_dbs)
```

Set the maximal number of doorbells that can be associated with the completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to modify. Must be stopped.

num_dbs

The maximal number of doorbells that can be associated with the context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_completion_start
(doca_devemu_pci_db_completion *db_comp)
```

Start DOCA devemu PCI device doorbell completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to start.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

On start verifies and finalizes the completion context configuration.

The following is possible for started completion context:

- ▶ Associating DOCA devemu PCI device doorbells with the completion context.

The following is NOT possible while completion context is started:

- ▶ Setting the properties of the completion context

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_db_completion_stop`
`(doca_devemu_pci_db_completion *db_comp)`

Stop DOCA devemu PCI device doorbell completion context.

Parameters

db_comp

The DOCA devemu PCI device doorbell completion context to stop.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

On stop prevents execution of different operations and allows operations that were available before start. For details, see [`doca_devemu_pci_db_completion_start\(\)`](#). Completion context can't be stopped while there are DOCA devemu PCI device doorbells associated with it.

The following is possible for stopped completion context:

- ▶ Setting the properties of the completion context

The following is NOT possible while completion context is stopped:

- ▶ Associating DOCA devemu PCI device doorbells with the completion context.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_create_on_dpa
(doca_devemu_pci_dev *pci_dev,
doca_devemu_pci_db_completion *db_comp, uint8_t
bar_id, uint64_t bar_start_addr, uint32_t db_id, uint64_t
user_data_on_dpa, doca_devemu_pci_db **db)
```

Allocate DOCA devemu PCI device doorbell on DPA. The created doorbell will be associated with a single completion context that was also created on DPA.

Parameters

pci_dev

The DOCA devemu PCI device to be associated with the doorbell. Must be started.

db_comp

The DOCA devemu PCI device doorbell completion context to be associated with the doorbell. Must be started.

bar_id

The identifier of the BAR that contains the associated doorbell region for the created doorbell.

bar_start_addr

The start address of the associated doorbell region within the BAR. This value must conform with the start address that was configured to the doorbell region during the configuration cycle of the PCI type that is associated with the given PCI device.

db_id

The doorbell identifier that will be used to map the doorbell to its handler. This value must be in the range of [0, num_db - 1] when num_db is the number of doorbells configured to the associated DOCA devemu PCI device. The default num_db value configured for any DOCA devemu PCI device created by [doca_devemu_pci_dev_create\(\)](#), if not configured otherwise, is equal to the value returned in [doca_devemu_pci_cap_type_get_max_num_db\(\)](#).

user_data_on_dpa

The user data that is associated with and can be retrieved by the DOCA devemu PCI device doorbell DPA handle.

db

The newly created DOCA devemu PCI device doorbell.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_destroy (doca_devemu_pci_db *db)
```

Destroy the DOCA devemu PCI device doorbell.

Parameters

db

The DOCA devemu PCI device doorbell to destroy. Must be stopped.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

If the doorbell was created on dpa, the associated dpa handle will be destroyed as well.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_get_dpa_handle
(doca_devemu_pci_db *db, doca_dpa_dev_devemu_pci_db_t
*db_handle)
```

Get the DPA handle for the DOCA devemu PCI device doorbell.

Parameters

db

The DOCA devemu PCI device doorbell previously created on DPA.

db_handle

A pointer to the associated DPA handle in the dpa memory space.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_db_modify_value (doca_devemu_pci_db
*db, uint32_t db_value)
```

Modify the current value of DOCA devemu PCI device doorbell. If the doorbell is started, this setting will behave as if the doorbell value was modified by the PCI device driver. If doorbell

value will not be modified before starting the DOCA devemu PCI device doorbell, the device will keep the current value of the doorbell.

Parameters

db

The DOCA devemu PCI device doorbell to modify.

db_value

The new value of the DOCA devemu PCI device doorbell.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_db_query_value (`doca_devemu_pci_db`
`*db, uint32_t *db_value`)

Query the current value of DOCA devemu PCI device doorbell.

Parameters

db

The DOCA devemu PCI device doorbell to query.

db_value

The current value of the DOCA devemu PCI device doorbell.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_db_start (`doca_devemu_pci_db *db`)

Start DOCA devemu PCI device doorbell. A started doorbell will be able to trigger completions on the associated doorbell completion context. Therefore, in case the doorbell was created on DPA, one should bind the associated doorbell handle to its doorbell completion context before starting the doorbell.

Parameters

db

The DOCA devemu PCI device doorbell to start.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_db_stop (doca_devemu_pci_db *db)

Stop DOCA devemu PCI device doorbell. A stopped doorbell will not trigger completions on the associated doorbell completion context. Therefore, in case the doorbell was created on DPA, one should stop the doorbell before un-binding the associated doorbell handle from its doorbell completion context.

Parameters

db

The DOCA devemu PCI device doorbell to stop.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

DOCA_EXPERIMENTAL doca_ctx *doca_devemu_pci_dev_as_ctx (doca_devemu_pci_dev *pci_dev)

Convert DOCA devemu PCI device instance into DOCA context.

Parameters

pci_dev

DOCA devemu PCI device. The device must remain valid until after the returned DOCA context is no longer required.

Returns

DOCA context upon success, NULL otherwise.

DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_dev_create (doca_devemu_pci_type *pci_type, doca_dev_rep *dev_rep, doca_pe *progress_engine, doca_devemu_pci_dev **pci_dev)

Allocate DOCA devemu PCI device.

Parameters

pci_type

The DOCA PCI type to be associated with the device. Must be started.

dev_rep

Representor DOCA device.

progress_engine

The progress engine that will be used to receive events and task completions.

pci_dev

The newly created DOCA devemu PCI device.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type', 'dev_rep', 'progress_engine' or 'pci_dev' are NULL. Or representor type does not match the PCI type
- ▶ DOCA_ERROR_NO_MEMORY - allocation failure

DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_dev_destroy (doca_devemu_pci_dev *pci_dev)

Free a DOCA devemu PCI device.

Parameters**pci_dev**

The previously created DOCA devemu PCI device. Must be idle.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' is NULL
- ▶ DOCA_ERROR_BAD_STATE - device is not idle. Use doca_ctx_stop() to stop it
- ▶ DOCA_ERROR_NOT_PERMITTED - PCI device was not created using doca_devemu_pci_dev_create()

DOCA_EXPERIMENTAL uint8_t doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get_b (doca_devemu_pci_dev_event_bar_stateful_region_driver_write *event)

Get the BAR id from BAR stateful region driver write event.

Parameters**event**

The registered BAR stateful region driver write event. Must not be NULL.

Returns

The BAR id that is associated with the event.

```
DOCA_EXPERIMENTAL uint64_t
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get_b
(doca_devemu_pci_dev_event_bar_stateful_region_driver_write
*event)
```

Get the BAR region start address that is associated with BAR stateful region driver write event.

Parameters

event

The registered BAR stateful region driver write event. Must not be NULL.

Returns

The start address of the BAR stateful region that is associated with the event.

```
DOCA_EXPERIMENTAL doca_devemu_pci_dev
*doca_devemu_pci_dev_event_bar_stateful_region_driver_write_get
(doca_devemu_pci_dev_event_bar_stateful_region_driver_write
*event)
```

Get DOCA devemu PCI device from BAR stateful region driver write event.

Parameters

event

The registered BAR stateful region driver write event. Must not be NULL.

Returns

The DOCA devemu PCI device associated with the event.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_regist
(doca_devemu_pci_dev *pci_dev,
doca_devemu_pci_dev_event_bar_stateful_region_driver_write_handl
handler, uint8_t bar_id, uint64_t bar_region_start_addr,
doca_data user_data)
```

Register to BAR stateful region driver write event.

Parameters

pci_dev

The DOCA devemu PCI device to be associated with the event. Must be idle.

handler

Method that is invoked once event is triggered.

bar_id

The BAR id to be associated with the event. Must conform with the BAR stateful region configuration that was done using [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#).

bar_region_start_addr

The start address of the BAR stateful region to be associated with the event. Must conform with the BAR stateful region configuration that was done using [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#).

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not idle

Description

Registration can be done only while DOCA devemu PCI device is idle. If called multiple times, for the same {pci_dev, bar_id, bar_region_start_addr} tuple, then only the last call will take effect. The registration will be valid for the entire stateful region that was configured for the associated DOCA devemu PCI device.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_event_flr_register
(doca_devemu_pci_dev *pci_dev,
doca_devemu_pci_dev_event_flr_handler_cb_t handler,
doca_data user_data)
```

Register to PCI FLR (Function Level Reset) event.

Parameters**pci_dev**

The DOCA devemu PCI device to be associated with the event. Must be idle.

handler

Method that is invoked once event is triggered.

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not idle

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_event_hotplug_state_change_register
(doca_devemu_pci_dev *pci_dev,
doca_devemu_pci_dev_event_hotplug_state_change_handler_cb_t
handler, doca_data user_data)
```

Register to hotplug state changes.

Parameters

pci_dev

The DOCA devemu PCI device to be associated with the event. Must be idle.

handler

Method that is invoked once event is triggered.

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not idle

Description

Registration can be done only while DOCA devemu PCI device is idle. If called multiple times then only the last call will take effect.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_get_hotplug_state
(doca_devemu_pci_dev *pci_dev,
doca_devemu_pci_hotplug_state **state)
```

Get the hotplug state of the DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device. Must be started.

state

The hotplug state of the given DOCA devemu PCI device.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' or 'state' are NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not started

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_hotplug (doca_devemu_pci_dev
*pci_dev)
```

Issue hotplug procedure of the DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device to hotplug. Must be started.

Returns

DOCA_SUCCESS - in case of success. On success, pci_dev is at DOCA_DEVEMU_PCI_HP_STATE_PLUG_IN_PROGRESS state. Event will not be raised in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' is NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not started, or hotplug state is not DOCA_DEVEMU_PCI_HP_STATE_POWER_OFF
- ▶ DOCA_ERROR_DRIVER - internal doca driver error
- ▶ DOCA_ERROR_NOT_SUPPORTED - The DOCA device that was set does not support hotplug, use doca_devemu_pci_cap_type_is_hotplug_supported() to find device that supports it

DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_dev_hotunplug (doca_devemu_pci_dev *pci_dev)

Issue hot unplug procedure of the DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device to hot unplug. Must be started.

Returns

DOCA_SUCCESS - in case of success. On success, pci_dev is at DOCA_DEVEMU_PCI_HP_STATE_UNPLUG_IN_PROGRESS state. Event will not be raised in case of success. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' is NULL
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not started, or current hotplug state is not one of DOCA_DEVEMU_PCI_HP_STATE_POWER_ON or DOCA_DEVEMU_PCI_HP_STATE_PLUG_IN_PROGRESS
- ▶ DOCA_ERROR_DRIVER - internal doca driver error
- ▶ DOCA_ERROR_NOT_SUPPORTED - The DOCA device that was set does not support hot unplug, use doca_devemu_pci_cap_type_is_hotplug_supported() to find device that supports it

DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_dev_is_flr (const doca_devemu_pci_dev *pci_dev, uint8_t *flr)

Query whether the DOCA devemu PCI device is having FLR (Function Level Reset).

Parameters

pci_dev

The DOCA devemu PCI device to query. Must be started.

flr

1 if the DOCA devemu PCI device is having FLR, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_dev' or 'flr' are NULL.
- ▶ DOCA_ERROR_BAD_STATE - PCI device is not started.

Description

If true, all PCI I/O transactions to/from the host memory are disabled and the user should re-configure the emulated PCI device. This re-configuration requires destruction of all the associated resources (e.g. DBs, MSIXs, MMAPPs), resetting the associated emulated PCI device (perform stop() and start() operations) and re-creating all the needed resources.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_modify_bar_stateful_region_default_values
 (doca_devemu_pci_dev *pci_dev, uint8_t id, uint64_t
 start_addr, void *default_values, uint64_t size)

Modify default registers values for stateful region in a DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device. Must be started.

id

The BAR id that contains the stateful region.

start_addr

The start address of the region within the BAR. This value must conform with the start address provided during [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#).

default_values

Input buffer that contain the default values data.

size

The size of the default_values buffer in bytes. The size must not be smaller than the actual size of the stateful bar region that was configured using [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#). If size is bigger than the actual size, the first relevant bytes will be used according to the actual size. The rest of the buffer will be ignored.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description

This method will modify the default values for the entire stateful region registers area in a PCI device BAR (before the first modification, the initial default values of the stateful region registers are taken from the associated PCI type). These values will override the previous default values and will become valid during the next exposure/hotplug of the associated PCI device to the host or during the next FLR.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_modify_bar_stateful_region_values
(doca_devemu_pci_dev *pci_dev, uint8_t id, uint64_t offset,
void *values, uint64_t size)
```

Modify registers values for stateful region in a DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device. Must be started.

id

The BAR id that contains the stateful region.

offset

The offset of the registers region to modify within the BAR. Must be located within the stateful BAR region.

values

Input buffer that contain the values data.

size

The size of the values buffer in bytes. The (offset + size) must be located within the stateful BAR region.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

This method will modify the values of the stateful region registers in a PCI device BAR. These values will override the existing values of the stateful region of the associated PCI device. Modifying registers by calling this method will not trigger the registered event handler of the `doca_devemu_pci_dev_event_bar_stateful_region_driver_write` event.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_query_bar_stateful_region_values
(doca_devemu_pci_dev *pci_dev, uint8_t id, uint64_t offset,
void *out_values, uint64_t size)
```

Query registers values of the stateful region in a DOCA devemu PCI device.

Parameters

pci_dev

The DOCA devemu PCI device. Must be started.

id

The BAR id that contains the stateful region.

offset

The offset of the registers region to query within the BAR. Must be located within the stateful BAR region.

out_values

Output buffer that will contain the values data upon success.

size

The size of the out_values buffer in bytes. The (offset + size) must be located within the stateful BAR region.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_mmap_create (doca_devemu_pci_dev`
`*pci_dev, doca_mmap **mmap)`

Allocates zero size memory map object with default/unset attributes associated with a DOCA devemu PCI device.

Parameters**pci_dev**

The DOCA devemu PCI device to be associated with the `doca_mmap`. Must be started.

mmap

DOCA memory map structure with default/unset attributes.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

The returned memory map object can be manipulated with common `doca_mmap` APIs.

The created memory map object will cover a memory range in the domain that hosts the DOCA devemu PCI device.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_msix_create_on_dpa
(doca_devemu_pci_dev *pci_dev, uint8_t bar_id,
uint64_t bar_start_addr, uint16_t msix_idx, uint64_t
user_data_on_dpa, doca_devemu_pci_msix **msix)
```

Allocate DOCA devemu PCI device MSI-X context on DPA.

Parameters

pci_dev

The DOCA devemu PCI device to be associated with the MSI-X. Must be started.

bar_id

The identifier of the BAR that contains the associated MSI-X table region for the created msix. This value must conform with the identifier that was configured to the MSI-X table region during the configuration cycle of the pci type that is associated with the given PCI device.

bar_start_addr

The start address of the associated MSI-X table region within the BAR. This value must conform with the start address that was configured to the MSI-X table region during the configuration cycle of the PCI type that is associated with the given PCI device.

msix_idx

The associated MSI-X table entry index.

user_data_on_dpa

The user data that is associated with and can be retrieved by the DOCA devemu PCI device MSI-X DPA handle.

msix

The newly created DOCA devemu PCI device MSI-X.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_msix_destroy (doca_devemu_pci_msix
*msix)
```

Destroy the DOCA devemu PCI device MSI-X.

Parameters

msix

The DOCA devemu PCI device MSI-X context to destroy.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

Description

If the MSI-X was created on dpa, the associated dpa handle will be destroyed as well.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_msix_get_dpa_handle
(doca_devemu_pci_msix *msix,
doca_dpa_dev_devemu_pci_msix_t *msix_handle)
```

Get the DPA handle for the DOCA devemu PCI device MSI-X.

Parameters

msix

The DOCA devemu PCI device MSI-X previously created on DPA.

msix_handle

A pointer to the associated DPA handle in the dpa memory space.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

2.10.1.1. DOCA Device Emulation - PCI Device Types

DOCA Device Emulation - PCI Devices

DOCA PCI type for emulated pci devices

enum doca_devemu_pci_bar_mem_type

According to the PCI specification a BAR that is mapped into Memory Space can define memory types.

Values

DOCA_DEVEMU_PCI_BAR_MEM_TYPE_32_BIT = 0

Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space. Base register support memory space below 1 MB

DOCA_DEVEMU_PCI_BAR_MEM_TYPE_1_MB

Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.

DOCA_DEVEMU_PCI_BAR_MEM_TYPE_64_BIT

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_db_region_get_max_num_region_blocks
(const doca_devinfo *devinfo, uint32_t *max_blocks)**

Get the maximum number of region blocks of a single doorbell BAR region that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it. The maximal number of region blocks together with the region block size defines the maximal size of a single doorbell region.

Parameters

devinfo

The device to query.

max_blocks

Maximal number of region blocks for a single doorbell BAR region that will be configured using `doca_devemu_pci_type_bar_db_region_<*>_conf_set()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_db_region_get_region_block_size
(const doca_devinfo *devinfo, uint32_t *block_size)**

Get the region block size of a doorbell BAR region that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it. The region block size is the smallest allocation data unit for a BAR region. For example, if the region block size is 64B then the bar region size can be 64B/128B/192B/.. / N*64B (N = max num region blocks per doorbell BAR region).

Parameters

devinfo

The device to query.

block_size

Region block size, in bytes, of a doorbell BAR region that will be configured using `doca_devemu_pci_type_bar_db_region_<*>_conf_set()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_cap_bar_db_region_get_start_addr_alignment
(const doca_devinfo *devinfo, uint32_t *alignment)

Get the doorbell BAR region start address alignment that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters

devinfo

The device to query.

alignment

The start address alignment, in bytes, of doorbell BAR regions that can be configured using `doca_devemu_pci_type_bar_db_region_<*>_conf_set()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_cap_bar_db_region_is_identify_by_data_supported
(const doca_devinfo *devinfo, uint8_t *supported)

Check if the device supports configuring doorbell regions that can identify doorbells by their data. If supported, one can configure a doorbell region using `doca_devemu_pci_type_set_bar_db_region_by_data_conf()`.

Parameters

devinfo

The device to query.

supported

1 if the doorbell region by data supported, 0 otherwise.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_cap_bar_db_region_is_identify_by_offset_supported
 (`const doca_devinfo *devinfo, uint8_t *supported`)

Check if the device supports configuring doorbell regions that can identify doorbells by their offset in the BAR. If supported, one can configure a doorbell region using `doca_devemu_pci_type_set_bar_db_region_by_offset_conf()`.

Parameters

devinfo

The device to query.

supported

1 if the doorbell region by offset supported, 0 otherwise.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_cap_bar_get_max_bar_db_regions (`const doca_devinfo *devinfo, uint32_t *max_regions`)

Get the maximum number of BAR doorbell regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`, per BAR.

Parameters

devinfo

The device to query.

max_regions

Number of BAR doorbell regions that can be configured per BAR, for any pci type, using `doca_devemu_pci_type_bar_db_region_<*>_conf_set()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_get_max_bar_msix_pba_regions (const
doca_devinfo *devinfo, uint32_t *max_regions)

Get the maximum number of BAR MSI-X PBA regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`, per BAR.

Parameters

devinfo

The device to query.

max_regions

Number of BAR MSI-X PBA regions that can be configured per BAR, for any pci type, using `doca_devemu_pci_type_set_bar_msix_pba_region_conf()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_get_max_bar_msix_table_regions
(const doca_devinfo *devinfo, uint32_t *max_regions)

Get the maximum number of BAR MSI-X table regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`, per BAR.

Parameters

devinfo

The device to query.

max_regions

Number of BAR MSI-X table regions that can be configured per BAR, for any pci type, using `doca_devemu_pci_type_set_bar_msix_table_region_conf()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_get_max_bar_regions (const
doca_devinfo *devinfo, uint32_t *max_bar_regions)
```

Get the maximum number of BAR regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`, per BAR.

Parameters

devinfo

The device to query.

max_bar_regions

Number of BAR regions that can be configured per BAR, for any pci type, using `doca_devemu_pci_type_bar_*_region_conf_set()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_get_max_bar_stateful_regions (const
doca_devinfo *devinfo, uint32_t *max_regions)
```

Get the maximum number of BAR stateful regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`, per BAR.

Parameters

devinfo

The device to query.

max_regions

Number of BAR stateful regions that can be configured per BAR, for any pci type, using `doca_devemu_pci_type_set_bar_stateful_region_conf()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_msix_pba_region_get_max_num_region_blocks
(const doca_devinfo *devinfo, uint32_t *max_blocks)
```

Get the maximum number of region blocks of a single MSI-X PBA BAR region that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be

created by it. The maximal number of region blocks together with the region block size defines the maximal size of a single MSI-X PBA region.

Parameters

devinfo

The device to query.

max_blocks

Maximal number of region blocks for a single MSI-X PBA BAR region that will be configured using `doca_devemu_pci_type_set_bar_msix_pba_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t` `doca_devemu_pci_cap_bar_msix_pba_region_get_region_block_size` (`const doca_devinfo *devinfo, uint32_t *block_size`)

Get the region block size of a MSI-X PBA BAR region that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it. The region block size is the smallest allocation data unit for a BAR region. For example, if the region block size is 64B then the bar region size can be 64B/128B/192B/.. / N*64B (N = max num region blocks per MSI-X PBA BAR region).

Parameters

devinfo

The device to query.

block_size

Region block size, in bytes, of a MSI-X PBA BAR region that will be configured using `doca_devemu_pci_type_set_bar_msix_pba_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t` `doca_devemu_pci_cap_bar_msix_pba_region_get_start_addr_alignment` (`const doca_devinfo *devinfo, uint32_t *alignment`)

Get the MSI-X PBA BAR region start address alignment that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters

devinfo

The device to query.

alignment

The start address alignment, in bytes, of MSI-X PBA BAR regions that can be configured using [doca_devemu_pci_type_set_bar_msix_pba_region_conf\(\)](#), for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL doca_error_t**doca_devemu_pci_cap_bar_msix_table_region_get_max_num_region_blocks (const doca_devinfo *devinfo, uint32_t *max_blocks)**

Get the maximum number of region blocks of a single MSI-X table BAR region that can be configured for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#) or will be created by it. The maximal number of region blocks together with the region block size defines the maximal size of a single MSI-X table region.

Parameters**devinfo**

The device to query.

max_blocks

Maximal number of region blocks for a single MSI-X table BAR region that will be configured using [doca_devemu_pci_type_set_bar_msix_table_region_conf\(\)](#), for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL doca_error_t**doca_devemu_pci_cap_bar_msix_table_region_get_region_block_size (const doca_devinfo *devinfo, uint32_t *block_size)**

Get the region block size of a MSI-X table BAR region that can be configured to an emulated PCI device, for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#) or will be created by it. The region block size is the smallest allocation data unit for a BAR region. For example, if the region block size is 64B then the bar region size can be 64B/128B/192B/.../N*64B (N = max num region blocks per MSI-X table BAR region).

Parameters**devinfo**

The device to query.

block_size

Region block size, in bytes, of a MSI-X table BAR region that will be configured using [doca_devemu_pci_type_set_bar_msix_table_region_conf\(\)](#), for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_srror_t](#).

DOCA_EXPERIMENTAL [doca_error_t](#)

[doca_devemu_pci_cap_bar_msix_table_region_get_start_addr_alignment](#) ([const doca_devinfo *devinfo](#), [uint32_t *alignment](#))

Get the MSI-X table BAR region start address alignment that can be configured for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#) or will be created by it.

Parameters

devinfo

The device to query.

alignment

The start address alignment, in bytes, of MSI-X table BAR regions that can be configured using [doca_devemu_pci_type_set_bar_msix_table_region_conf\(\)](#), for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL [doca_error_t](#)

[doca_devemu_pci_cap_bar_stateful_region_get_max_num_region_blocks](#) ([const doca_devinfo *devinfo](#), [uint32_t *max_blocks](#))

Get the maximum number of region blocks of a single stateful BAR region that can be configured for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#) or will be created by it. The maximal number of region blocks together with the region block size defines the maximal size of a single stateful region.

Parameters

devinfo

The device to query.

max_blocks

Maximal number of region blocks for a single stateful BAR region that will be configured using [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#), for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_stateful_region_get_region_block_size
(const doca_devinfo *devinfo, uint32_t *block_size)

Get the region block size of a stateful BAR region that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it. The region block size is the smallest allocation data unit for a BAR region. For example, if the region block size is 64B then the bar region size can be 64B/128B/192B/.. / N*64B (N = max num region blocks per stateful BAR region).

Parameters

devinfo

The device to query.

block_size

Region block size, in bytes, of a stateful BAR region that will be configured using `doca_devemu_pci_type_set_bar_stateful_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_bar_stateful_region_get_start_addr_alignment
(const doca_devinfo *devinfo, uint32_t *alignment)

Get the stateful BAR region start address alignment that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters

devinfo

The device to query.

alignment

The start address alignment, in bytes, of stateful BAR regions that can be configured using `doca_devemu_pci_type_set_bar_stateful_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_get_max_types (const doca_devinfo *devinfo,
uint16_t *max_pci_types)
```

Get the maximum number of PCI types that can be created by the device.

Parameters

devinfo

The device to query.

max_pci_types

Number of PCI types that can be created using [doca_devemu_pci_type_create\(\)](#).

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_log_max_bar_size (const
doca_devinfo *devinfo, uint8_t *log_bar_size)
```

Get the maximum BAR size (in Log base 2) that can be configured for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#).

Parameters

devinfo

The device to query.

log_bar_size

The maximal BAR size, given in bytes, of single BAR in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_log_max_db_size (const
doca_devinfo *devinfo, uint8_t *log_db_size)
```

Get the maximum doorbell size (in Log base 2) that can be configured for any PCI type that was created using [doca_devemu_pci_type_create\(\)](#).

Parameters

devinfo

The device to query.

log_db_size

The maximal doorbell size, given in bytes, of single doorbell in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_cap_type_get_log_max_db_stride_size (const`
`doca_devinfo *devinfo, uint8_t *log_stride_size)`

Get the maximal stride size (in Log base 2) of a single doorbell that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

log_stride_size

The maximal single doorbell stride size, given in bytes, of single doorbell in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_cap_type_get_log_min_bar_size (const`
`doca_devinfo *devinfo, uint8_t *log_bar_size)`

Get the minimal BAR size (in Log base 2) that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

log_bar_size

The minimal BAR size, given in bytes, of single BAR in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_log_min_db_size (const
doca_devinfo *devinfo, uint8_t *log_db_size)
```

Get the minimal doorbell size (in Log base 2) that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

log_db_size

The minimal doorbell size, given in bytes, of single doorbell in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_log_min_db_stride_size (const
doca_devinfo *devinfo, uint8_t *log_stride_size)
```

Get the minimal stride size (in Log base 2) of a single doorbell that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

log_stride_size

The minimal single doorbell stride size, given in bytes, of single doorbell in Log (base 2) units.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_bar_db_regions (const
doca_devinfo *devinfo, uint32_t *max_regions)
```

Get the maximum amount of doorbell BAR regions that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters

devinfo

The device to query.

max_regions

Maximal number of doorbell BAR regions that can be configured using `doca_devemu_pci_type_bar_db_region_<*>_conf_set()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t

`doca_devemu_pci_cap_type_get_max_bar_msix_pba_regions (const doca_devinfo *devinfo, uint32_t *max_regions)`

Get the maximum amount of MSI-X PBA BAR regions that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters**devinfo**

The device to query.

max_regions

Maximal number of MSI-X PBA BAR regions that can be configured using `doca_devemu_pci_type_set_bar_msix_pba_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t

`doca_devemu_pci_cap_type_get_max_bar_msix_table_regions (const doca_devinfo *devinfo, uint32_t *max_regions)`

Get the maximum amount of MSI-X table BAR regions that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters**devinfo**

The device to query.

max_regions

Maximal number of MSI-X table BAR regions that can be configured using `doca_devemu_pci_type_set_bar_msix_table_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_bar_regions (const
doca_devinfo *devinfo, uint32_t *max_bar_regions)
```

Get the maximum number of BAR regions that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

max_bar_regions

Number of BAR regions that can be configured, for any pci type, using `doca_devemu_pci_type_bar_*_region_conf_set()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_bar_stateful_regions (const
doca_devinfo *devinfo, uint32_t *max_regions)
```

Get the maximum amount of stateful BAR regions that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` or will be created by it.

Parameters

devinfo

The device to query.

max_regions

Maximal number of stateful BAR regions that can be configured using `doca_devemu_pci_type_set_bar_stateful_region_conf()`, for any pci type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_bars (const doca_devinfo
*devinfo, uint8_t *max_bars)
```

Get the maximum number of BARs that can be configured to an emulated PCI device, for any PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

max_bars

Number of BARs that can be configured for any pci type using `doca_devemu_pci_type_*_bar_conf_set()`.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_num_db (const doca_devinfo
*devinfo, uint16_t *num_db)
```

Get the maximal number of doorbells that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` and for any PCI device that is associated with a PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

num_db

The maximal number of doorbells.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_get_max_num_msix (const
doca_devinfo *devinfo, uint16_t *num_msix)

Get the maximal number of MSIXs that can be configured for any PCI type that was created using `doca_devemu_pci_type_create()` and for any PCI device that is associated with a PCI type that was created using `doca_devemu_pci_type_create()`.

Parameters

devinfo

The device to query.

num_msix

The maximal number of MSIXs.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_is_bar_mem_type_supported
(const doca_devinfo *devinfo, doca_devemu_pci_bar_mem_type
memory_type, uint8_t *supported)

Get the memory BAR types capability of the device. If supported, A BAR with that memory type can be configured using `doca_devemu_pci_type_set_memory_bar_conf()`.

Parameters

devinfo

The device to query.

memory_type

The BAR memory type to query.

supported

1 if the BAR memory type is supported by the device, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'devinfo' or 'supported' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_is_hotplug_supported (const
doca_devinfo *devinfo, const doca_devemu_pci_type *pci_type,
uint8_t *supported)
```

Get the hotplug capability of the device for a given pci type.

Parameters

devinfo

The device to query.

pci_type

The DOCA pci type to query.

supported

1 if the hotplug capability is supported for this type, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'devinfo', 'pci_type' or 'supported' are NULL or devinfo doesn't match associated doca_dev, in case doca_dev already was assigned to the pci_type
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description

Get uint8_t value defining if the device can be used to hotplug devices for a specific type. The hotplug capability of a device implies its management capability.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_cap_type_is_mgmt_supported (const
doca_devinfo *devinfo, const doca_devemu_pci_type *pci_type,
uint8_t *supported)
```

Get the management capability of the device for a given pci type.

Parameters

devinfo

The device to query.

pci_type

The DOCA pci type to query.

supported

1 if the management capability is supported for this type, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'devinfo', 'pci_type' or 'supported' are NULL or devinfo doesn't match associated doca_dev, in case doca_dev already was assigned to the pci_type
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description

Get uint8_t value defining if the device can be used to manage devices for a specific type. The management capability of a device doesn't imply its hotplug capability.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_create_rep (const doca_devemu_pci_type
*pci_type, doca_dev_rep **dev_rep)
```

Create a new representor device for a given pci type.

Parameters

pci_type

The DOCA pci type. Must be started.

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 - 'pci_type' or 'dev_rep' are NULL
- ▶ DOCA_ERROR_BAD_STATE - type is not started
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description

The representor device will not be visible to the host. The representor device will be discoverable using the discovery mechanism for representors.



Note:

pci type must be started.

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_dev_destroy_rep (doca_dev_rep *rep_dev)**

Destroy a representor device that was created using `doca_devemu_pci_dev_create_rep()`.

Parameters

rep_dev

Previously initialized representor device instance.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'rep_dev' is NULL or corrupted
- ▶ DOCA_ERROR_NOT_PERMITTED - representor was not created using `doca_devemu_pci_dev_create_rep()`
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description

The representor device will not be visible to the host. The representor device will not be discoverable using the discovery mechanism for representors.

**DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_type_create
(const char *name, doca_devemu_pci_type **pci_type)**

Create a stopped DOCA devemu pci type.

Parameters

name

The name to assign to the created DOCA devemu pci type. The NULL terminated string must not exceed `DOCA_DEVEMU_PCI_TYPE_NAME_LEN`.

pci_type

The created and stopped DOCA devemu pci type.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'name' or 'pci_type' are NULL
- ▶ DOCA_ERROR_NO_MEMORY - allocation failure

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_type_create_rep_list (`doca_devemu_pci_type`
`*pci_type, doca_devinfo_repdev_list_rep, uint32_t *nb_devs_rep`)
 Create list of available representor devices for a given devemu pci type.

Parameters

pci_type

The DOCA devemu pci type. Must be started.

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. see `doca_error_t`.

Description

Returns all representors that are associated with the provided devemu pci type family. The provided pci type must be started. A representor can either represent an emulated PCI function that is currently attached to the host PCI subsystem or an emulated PCI function intended for hotplugging into the host PCI subsystem.



Note:

Returned list must be destroyed using [doca_devinfo_rep_destroy_list\(\)](#)

DOCA_EXPERIMENTAL `doca_error_t`
doca_devemu_pci_type_destroy (`doca_devemu_pci_type *pci_type`)
 Destroy a DOCA devemu pci type.

Parameters

pci_type

The DOCA devemu pci type to destroy. Must be stopped.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - type can't be destroyed while started. Need to stop it first - `doca_devemu_pci_type_stop()`

- ▶ DOCA_ERROR_NOT_PERMITTED - type was not created using `doca_devemu_pci_type_create()`

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_type_get_class_code (const`
`doca_devemu_pci_type *pci_type, uint32_t *class_code)`

Get the PCI Class Code of a pci type to identify generic operation.

Parameters

pci_type

The DOCA pci type to query.

class_code

The PCI Class Code to identify generic operation. Only 24 LSBits are valid.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'class_code' are NULL

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_type_get_dev (const doca_devemu_pci_type`
`*pci_type, doca_dev **dev)`

Get the doca device of a pci type.

Parameters

pci_type

The DOCA devemu pci type to query.

dev

DOCA dev.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'dev' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_device_id (const
doca_devemu_pci_type *pci_type, uint16_t *device_id)
```

Get the PCI Device ID of a pci type.

Parameters

pci_type

The DOCA pci type to query.

device_id

The PCI Device ID (DID) assigned by the vendor.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'device_id' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_name (const doca_devemu_pci_type
*pci_type, char(*) name)
```

Get the name of a pci type.

Parameters

pci_type

The DOCA pci type to query.

name

The name of the pci type.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'name' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_num_msix (const
doca_devemu_pci_type *pci_type, uint16_t *num_msix)
```

Get the size of the MSI-X Table from MSI-X Capability Registers (1 based) of a pci type. This value will be used as the default num_msix value for associated DOCA devemu pci devices, unless configured otherwise.

Parameters

pci_type

The DOCA pci type to query.

num_msix

The size of the MSI-X Table from MSI-X Capability Registers (1 based).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'num_msix' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_revision_id (const
doca_devemu_pci_type *pci_type, uint8_t *revision_id)
```

Get the PCI Revision ID of a pci type.

Parameters

pci_type

The DOCA pci type to query.

revision_id

The PCI Revision ID assigned by the vendor.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'revision_id' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_subsystem_id (const
doca_devemu_pci_type *pci_type, uint16_t *subsystem_id)
```

Get the PCI Subsystem ID of a pci type.

Parameters

pci_type

The DOCA pci type to query.

subsystem_id

The PCI Subsystem ID (SSID) assigned by the subsystem vendor.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'subsystem_id' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_subsystem_vendor_id (const
doca_devemu_pci_type *pci_type, uint16_t *subsystem_vid)
```

Get the PCI Subsystem Vendor ID of a pci type.

Parameters

pci_type

The DOCA pci type to query.

subsystem_vid

The PCI Subsystem Vendor ID (SVID) allocated by the PCI-SIG.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'subsystem_vid' are NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_get_vendor_id (const
doca_devemu_pci_type *pci_type, uint16_t *vendor_id)
```

Get the PCI Vendor ID of a pci type.

Parameters

pci_type

The DOCA pci type to query.

vendor_id

The PCI Vendor ID (VID) allocated by the PCI-SIG.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'vendor_id' are NULL

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_is_started (const doca_devemu_pci_type *pci_type, uint8_t *started)

Check whether the pci type is started.

Parameters**pci_type**

The DOCA pci type to query.

started

1 if the pci_type is started, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'started' are NULL.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_modify_bar_stateful_region_default_values (doca_devemu_pci_type *pci_type, uint8_t id, uint64_t start_addr, void *default_values, uint64_t size)

Modify default registers values for a configured stateful region in a DOCA devemu pci type.

Parameters**pci_type**

The DOCA devemu pci type. Must be started and must not be associated to any pci_dev or doca_dev_rep.

id

The BAR id that contains the stateful region. This value must conform with the id provided during [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#).

start_addr

The start address of the region within the BAR. This value must conform with the start address provided during [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#).

default_values

Input buffer that contain the default values data.

size

The size of the default_values buffer in bytes. The size must not be smaller than the actual size of the stateful bar region that was configured using [doca_devemu_pci_type_set_bar_stateful_region_conf\(\)](#). If size is bigger than the actual size, the first relevant bytes will be used according to the actual size. The rest of the buffer will be ignored.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description

This method will modify the default values for the entire stateful region registers area in a pci type BAR (before the first modification, the initial default values of the stateful region registers are zeroed). The [doca_dev_rep](#) that will be created using [doca_devemu_pci_dev_create_rep\(\)](#) from the associated [pci_type](#) will inherit these default values upon creation. Furthermore, each [pci_dev](#) that will be created from the associated [doca_dev_rep](#) will have a chance to update its own default values by calling [doca_devemu_pci_dev_modify_bar_stateful_region_default_values\(\)](#) and not effecting the default values of the [pci_type](#).

DOCA_EXPERIMENTAL [doca_error_t](#)
[doca_devemu_pci_type_set_bar_db_region_by_data_conf](#)
 ([doca_devemu_pci_type *pci_type](#), [uint8_t id](#), [uint64_t start_addr](#),
[uint64_t size](#), [uint8_t log_db_size](#), [uint16_t db_id_msbyte](#), [uint16_t db_id_lsbyte](#))

Set a doorbell BAR region configuration for a BAR layout in a DOCA devemu pci type. The doorbells that will be associated with this doorbell BAR region will be identified according to the data written to the doorbell. The doorbell identifier will be deduced from the written doorbell data by the [db_id_lsbyte](#) and [db_id_msbyte](#) settings.

Parameters**pci_type**

The DOCA devemu pci type. Must not be started.

id

The BAR id that will contain the new region.

start_addr

The start address of the region within the BAR. This value must conform with the start address alignment capability from [doca_devemu_pci_cap_bar_db_region_get_start_addr_alignment\(\)](#).

size

The size of the region in bytes. Must conform with [doca_devemu_pci_cap_bar_db_region_get_region_block_size\(\)](#) and [doca_devemu_pci_cap_bar_db_region_get_max_num_region_blocks\(\)](#).

log_db_size

The size, given in bytes, of single doorbell in Log (base 2) units. This value must conform with [doca_devemu_pci_cap_type_get_log_min_db_size\(\)](#) and [doca_devemu_pci_cap_type_get_log_max_db_size\(\)](#).

db_id_msbyte

The start byte of the doorbell identifier, within the doorbell data written by the driver. If the `db_id_msbyte > db_id_lsbyte` then the doorbell identifier will be treated as Little-Endian.

db_id_lsbyte

The end byte of the doorbell identifier, within the doorbell data written by the driver. If the `db_id_msbyte > db_id_lsbyte` then the doorbell identifier will be treated as Little-Endian.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL doca_error_t

doca_devemu_pci_type_set_bar_db_region_by_offset_conf
(`doca_devemu_pci_type *pci_type, uint8_t id, uint64_t start_addr, uint64_t size, uint8_t log_db_size, uint8_t log_stride_size`)

Set a doorbell BAR region configuration for a BAR layout in a DOCA devemu pci type. The doorbells that will be associated with this doorbell BAR region will be identified according to their offset within the BAR.

Parameters**pci_type**

The DOCA devemu pci type. Must not be started.

id

The BAR id that will contain the new region.

start_addr

The start address of the region within the BAR. This value must conform with the start address alignment capability from [doca_devemu_pci_cap_bar_db_region_get_start_addr_alignment\(\)](#).

size

The size of the region in bytes. Must conform with [doca_devemu_pci_cap_bar_db_region_get_region_block_size\(\)](#) and [doca_devemu_pci_cap_bar_db_region_get_max_num_region_blocks\(\)](#).

log_db_size

The size, given in bytes, of single doorbell in Log (base 2) units. This value must conform with [doca_devemu_pci_cap_type_get_log_min_db_size\(\)](#) and [doca_devemu_pci_cap_type_get_log_max_db_size\(\)](#).

log_stride_size

The size, given in bytes, of a single doorbell stride in Log (base 2) units. This value must conform with [doca_devemu_pci_cap_type_get_log_min_db_stride_size\(\)](#) and [doca_devemu_pci_cap_type_get_log_max_db_stride_size\(\)](#).

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL doca_error_t

doca_devemu_pci_type_set_bar_msix_pba_region_conf
([doca_devemu_pci_type](#) *pci_type, uint8_t id, uint64_t start_addr, uint64_t size)

Set a MSI-X PBA BAR region configuration for a BAR layout in a DOCA devemu pci type.

Parameters**pci_type**

The DOCA devemu pci type. Must not be started.

id

The BAR id that will contain the new region.

start_addr

The start address of the region within the BAR. This value must conform with the start address alignment capability from [doca_devemu_pci_cap_bar_msix_pba_region_get_start_addr_alignment\(\)](#).

size

The size of the region in bytes. Must correlate with the num_msix value of the pci type and conform with [doca_devemu_pci_cap_bar_msix_pba_region_get_region_block_size\(\)](#) and [doca_devemu_pci_cap_bar_msix_pba_region_get_max_num_region_blocks\(\)](#).

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

DOCA_EXPERIMENTAL doca_error_t

doca_devemu_pci_type_set_bar_msix_table_region_conf
([doca_devemu_pci_type](#) *pci_type, uint8_t id, uint64_t start_addr, uint64_t size)

Set a MSI-X table BAR region configuration for a BAR layout in a DOCA devemu pci type.

Parameters**pci_type**

The DOCA devemu pci type. Must not be started.

id

The BAR id that will contain the new region.

start_addr

The start address of the region within the BAR. This value must conform with the start address alignment capability from [doca_devemu_pci_cap_bar_msix_table_region_get_start_addr_alignment\(\)](#).

size

The size of the region in bytes. Must correlate with the num_msix value of the pci type and conform with [doca_devemu_pci_cap_bar_msix_table_region_get_region_block_size\(\)](#) and [doca_devemu_pci_cap_bar_msix_table_region_get_max_num_region_blocks\(\)](#).

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_bar_stateful_region_conf
(doca_devemu_pci_type *pci_type, uint8_t id, uint64_t start_addr,
uint64_t size)**

Set a stateful BAR region configuration for a BAR layout in a DOCA devemu pci type.

Parameters**pci_type**

The DOCA devemu pci type. Must not be started.

id

The BAR id that will contain the new region.

start_addr

The start address of the region within the BAR. This value must conform with the start address alignment capability from [doca_devemu_pci_cap_bar_stateful_region_get_start_addr_alignment\(\)](#).

size

The size of the region in bytes. Must conform with [doca_devemu_pci_cap_bar_stateful_region_get_region_block_size\(\)](#) and [doca_devemu_pci_cap_bar_stateful_region_get_max_num_region_blocks\(\)](#).

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description

The stateful BAR region will be used by the device to expose registers to the driver. This region will be actively maintained by the device. PCI READ transactions for this region will be automatically answered by the device. PCI WRITE transactions will be stored to this region by the device. The associated pci_dev will be notified upon

any PCI WRITE transaction to this region initiated by the device driver using the `doca_devemu_pci_dev_event_bar_stateful_region_driver_write` event, if this event was register.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_type_set_class_code (doca_devemu_pci_type *pci_type, uint32_t class_code)`

Set the PCI Class Code of a pci type to identify generic operation.

Parameters

pci_type

The DOCA pci context to modify. Must not be started.

class_code

The PCI Class Code to identify generic operation. Only 24 LSBits are valid.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_pci_type_set_dev (doca_devemu_pci_type *pci_type, doca_dev *dev)`

Set the doca device of a pci type.

Parameters

pci_type

The DOCA devemu pci type to modify. Must not be started.

dev

DOCA dev.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' or 'dev' are NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it
- ▶ DOCA_ERROR_NOT_PERMITTED - can't set a device for default pci types

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_device_id (doca_devemu_pci_type
*pci_type, uint16_t device_id)
```

Set the PCI Device ID of a pci type.

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

device_id

The PCI Device ID (DID) to assign.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_io_bar_conf (doca_devemu_pci_type
*pci_type, uint8_t id, uint8_t log_sz)
```

Set an IO BAR layout configuration for DOCA devemu pci type.

Parameters

pci_type

The DOCA devemu pci type. Must not be started.

id

The BAR id.

log_sz

The BAR size, in Log (base 2) units. Must be set to 0 if previous BAR requires an extension (e.g. for 64-bit BARs). Otherwise, this value must conform with [doca_devemu_pci_cap_type_get_log_min_bar_size\(\)](#) and [doca_devemu_pci_cap_type_get_log_max_bar_size\(\)](#) capabilities.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description



Note:

This configuration is applicable only for type created by [doca_devemu_pci_type_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_memory_bar_conf
(doca_devemu_pci_type *pci_type, uint8_t id, uint8_t log_sz,
doca_devemu_pci_bar_mem_type memory_type, uint8_t
prefetchable)
```

Set a memory BAR layout configuration for DOCA devemu pci type.

Parameters

pci_type

The DOCA devemu pci type. Must not be started.

id

The BAR id.

log_sz

The BAR size, in Log (base 2) units. Must be set to 0 if previous BAR requires an extension (e.g. for 64-bit BARs). Otherwise, this value must conform with [doca_devemu_pci_cap_type_get_log_min_bar_size\(\)](#) and [doca_devemu_pci_cap_type_get_log_max_bar_size\(\)](#) capabilities.

memory_type

Memory type value to expose for this BAR.

prefetchable

Prefetchable bit value to expose for this BAR. Set to 1 if the BAR does not contain locations with side effects on reads. Set to 0 if the BAR contains locations with read side effects or locations in which the function does not tolerate write merging.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see [doca_error_t](#).

Description



Note:

This configuration is applicable only for type created by [doca_devemu_pci_type_create\(\)](#)

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_num_msix (doca_devemu_pci_type
 *pci_type, uint16_t num_msix)

Set the size of the MSI-X Table from MSI-X Capability Registers (1 based) of a pci type. Must conform with doca_devemu_pci_cap_type_get_max_num_msix().

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

num_msix

The size of the MSI-X Table from MSI-X Capability Registers (1 based).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_revision_id (doca_devemu_pci_type
 *pci_type, uint8_t revision_id)

Set the PCI Revision ID of a pci type.

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

revision_id

The PCI Revision ID to assign.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_subsystem_id (doca_devemu_pci_type
*pci_type, uint16_t subsystem_id)
```

Set the PCI Subsystem ID of a pci type.

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

subsystem_id

The PCI Subsystem ID (SSID) to assign.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_subsystem_vendor_id
(doca_devemu_pci_type *pci_type, uint16_t subsystem_vid)
```

Set the PCI Subsystem Vendor ID of a pci type.

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

subsystem_vid

The PCI Subsystem Vendor ID (SVID) allocated by the PCI-SIG.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_pci_type_set_vendor_id (doca_devemu_pci_type
*pci_type, uint16_t vendor_id)
```

Set the PCI Vendor ID of a pci type.

Parameters

pci_type

The DOCA pci type to modify. Must not be started.

vendor_id

The PCI Vendor ID (VID) allocated by the PCI-SIG.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - can't configure type after starting it

```
DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_type_start
(doca_devemu_pci_type *pci_type)
```

Start a DOCA devemu pci type.

Parameters

pci_type

The DOCA devemu pci type to start.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL or invalid PCI parameters were previously provided
- ▶ DOCA_ERROR_BAD_STATE - type is already started
- ▶ DOCA_ERROR_NOT_FOUND - device was not provided - use doca_devemu_pci_type_set_dev()
- ▶ DOCA_ERROR_NO_MEMORY - allocation failure
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description



Note:

This method upon success disable the ability to configure the DOCA devemu pci type.

`DOCA_EXPERIMENTAL doca_error_t doca_devemu_pci_type_stop
(doca_devemu_pci_type *pci_type)`

Stop a DOCA devemu pci type.

Parameters

pci_type

The DOCA devemu pci type to stop. Must not be associated to any pci device.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'pci_type' is NULL
- ▶ DOCA_ERROR_BAD_STATE - type is already stopped
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description



Note:

This method upon success re-enable the ability to configure the DOCA devemu pci type.

`#define DOCA_DEVEMU_PCI_TYPE_NAME_LEN 32`

Maximal length for the NULL terminated string that describe the name of the emulated PCI device type.

2.10.2. DOCA Device Emulation - Virtio FS Devices

DOCA Device Emulation

DOCA library for emulated virtio FS devices

[DOCA Device Emulation - Virtio FS IO Context](#)

[DOCA Device Emulation - Virtio FS Device Types](#)

```
DOCA_EXPERIMENTAL doca_ctx
*doca_devemu_vfs_dev_as_ctx (doca_devemu_vfs_dev
*vfs_dev)
```

Convert DOCA Virtio FS device instance into DOCA context.

Parameters

vfs_dev

DOCA Virtio FS device instance. This must remain valid until after the DOCA context is no longer required.

Returns

doca ctx upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_devemu_pci_dev
*doca_devemu_vfs_dev_as_pci_dev (doca_devemu_vfs_dev
*vfs_dev)
```

Convert DOCA Virtio FS device instance into DOCA devemu PCI device.

Parameters

vfs_dev

DOCA Virtio FS device instance. This must remain valid until after the DOCA devemu PCI device is no longer required.

Returns

DOCA devemu pci device upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_devemu_virtio_dev
*doca_devemu_vfs_dev_as_virtio_dev
(doca_devemu_vfs_dev *vfs_dev)
```

Convert DOCA Virtio FS device instance into DOCA Virtio device.

Parameters

vfs_dev

DOCA Virtio FS device instance. This must remain valid until after the DOCA Virtio device is no longer required.

Returns

doca virtio device upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_create (doca_devemu_vfs_type
*vfs_type, doca_dev_rep *dev_rep, doca_pe
*progress_engine, doca_devemu_vfs_dev **vfs_dev)
```

Allocate DOCA Virtio FS device.

Parameters

vfs_type

The DOCA Virtio FS type to be associated to the device. Must be started.

dev_rep

Representor DOCA device.

progress_engine

The progress engine that will be used to receive events and task completions.

vfs_dev

The newly created DOCA Virtio FS device.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_destroy (doca_devemu_vfs_dev
*vfs_dev)
```

Free a DOCA Virtio FS device object.

Parameters

vfs_dev

The previously created DOCA Virtio FS device.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_get_notify_buf_size (const
doca_devemu_vfs_dev *vfs_dev, uint32_t *notify_buf_size)
```

Get the value of the VIRTIO FS Device notify_buf_size register.

Parameters

vfs_dev

The DOCA Virtio FS device instance to query.

notify_buf_size

The value of virtio_fs_config:notify_buf_size register according to virtio specification.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'notify_buf_size' is NULL

Description

The notify_buf_size value will be used only if VIRTIO_FS_F_NOTIFICATION feature bit is set. Therefore, the notify_buf_size value must comply with VIRTIO_FS_F_NOTIFICATION feature bit before starting the associated vfs_dev.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_get_num_request_queues
(const doca_devemu_vfs_dev *vfs_dev, uint32_t
*num_request_queues)
```

Get the value of the VIRTIO FS Device num_request_queues register.

Parameters

vfs_dev

The DOCA Virtio FS device instance to query.

num_request_queues

The value of Device virtio_fs_config:num_request_queues register according to virtio specification.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'num_request_queues' is NULL

DOCA_EXPERIMENTAL doca_error_t doca_devemu_vfs_dev_get_tag (const doca_devemu_vfs_dev *vfs_dev, char tag)

Get the value of the Virtio FS device tag. According to the specification the tag is encoded in UTF-8 and padded with NULL bytes if shorter than the available space of 36 bytes and is not NULL-terminated if the encoded bytes take up the entire field of 36 bytes. In DOCA, the tag is always NULL terminated and is shorter than the Virtio specification definition.

Parameters

vfs_dev

The DOCA Virtio FS device instance to query.

tag

The value of the Virtio FS Device virtio_fs_config:tag according to Virtio specification (with NULL termination).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'tag' is NULL

DOCA_EXPERIMENTAL doca_error_t doca_devemu_vfs_dev_get_vfs_notification_req_user_data_size (const doca_devemu_vfs_dev *vfs_dev, uint32_t *req_user_data_size)

Get the size of the user data buffer that will be allocated for each doca_devemu_vfs_notification_req on behalf of the user. This buffer will be valid and used by the user upon receiving new doca_devemu_vfs_notification_req. The buffer will become invalid after doca_devemu_vfs_notification_req completion.

Parameters

vfs_dev

The DOCA Virtio FS device instance to query.

req_user_data_size

Size, in bytes, of the user data buffer to be allocated on behalf of the user for each doca_devemu_vfs_notification_req.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'req_user_data_size' is NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_get_vfs_req_user_data_size
(const doca_devemu_vfs_dev *vfs_dev, uint32_t
*req_user_data_size)
```

Get the size of the user data buffer that will be allocated for each `doca_devemu_vfs_req` on behalf of the user. This buffer will be valid and used by the user upon receiving new `doca_devemu_vfs_req`. The buffer will become invalid after `doca_devemu_vfs_req` completion.

Parameters

vfs_dev

The DOCA Virtio FS device instance to query.

req_user_data_size

Size, in bytes, of the user data buffer to be allocated on behalf of the user for each `doca_devemu_vfs_req`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'req_user_data_size' is NULL

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_set_notify_buf_size
(doca_devemu_vfs_dev *vfs_dev, uint32_t notify_buf_size)
```

Set the value of the VIRTIO FS Device `notify_buf_size` register.

Parameters

vfs_dev

The DOCA Virtio FS device instance to modify. Must be idle.

notify_buf_size

The value of `virtio_fs_config:notify_buf_size` register according to virtio specification.

This value must be power of 2 if the `VIRTIO_FS_F_NOTIFICATION` bit is set. If the `VIRTIO_FS_F_NOTIFICATION` feature bit is unset, a value of 0 can be used. The compliance between `notify_buf_size` and `VIRTIO_FS_F_NOTIFICATION` will be verified upon starting the associated `vfs_dev`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' is NULL or `notify_buf_size` value is invalid.
- ▶ DOCA_ERROR_BAD_STATE - device is not idle

Description

The `notify_buf_size` value will be used only if `VIRTIO_FS_F_NOTIFICATION` feature bit is set. Therefore, the `notify_buf_size` value must comply with `VIRTIO_FS_F_NOTIFICATION` feature bit before starting the associated `vfs_dev`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_set_num_request_queues
(doca_devemu_vfs_dev *vfs_dev, uint32_t
num_request_queues)
```

Set the value of the VIRTIO FS Device `num_request_queues` register.

Parameters

vfs_dev

The DOCA Virtio FS device instance to modify. Must be idle.

num_request_queues

The value of Device `virtio_fs_config:num_request_queues` register according to virtio specification.

Returns

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

- ▶ `DOCA_ERROR_INVALID_VALUE` - '`vfs_dev`' is NULL
- ▶ `DOCA_ERROR_BAD_STATE` - device is not idle

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_set_tag (doca_devemu_vfs_dev
*vfs_dev, const char tag)
```

Set the value of the Virtio FS device tag. According to the specification the tag is encoded in UTF-8 and padded with NULL bytes if shorter than the available space of 36 bytes and is not NULL-terminated if the encoded bytes take up the entire field of 36 bytes. In DOCA, the tag is always NULL terminated and is shorter than the Virtio specification definition.

Parameters

vfs_dev

The DOCA Virtio FS device instance to modify. Must be idle.

tag

The value of the Virtio FS Device `virtio_fs_config:tag` according to Virtio specification (with NULL termination).

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' or 'tag' is NULL
- ▶ DOCA_ERROR_BAD_STATE - device is not idle
- ▶ DOCA_ERROR_TOO_BIG - tag is greater than 20 bytes

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_set_vfs_notification_req_user_data_size
(doca_devemu_vfs_dev *vfs_dev, uint32_t
req_user_data_size)

Set the size of the user data buffer that will be allocated for each `doca_devemu_vfs_notification_req` on behalf of the user. This buffer will be valid and used by the user upon receiving new `doca_devemu_vfs_notification_req`. The buffer will become invalid after `doca_devemu_vfs_notification_req` completion.

Parameters

vfs_dev

The DOCA Virtio FS device instance to modify. Must be idle.

req_user_data_size

Size, in bytes, of the user data buffer to be allocated on behalf of the user for each `doca_devemu_vfs_notification_req`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' is NULL
- ▶ DOCA_ERROR_BAD_STATE - device is not idle

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_dev_set_vfs_req_user_data_size
(doca_devemu_vfs_dev *vfs_dev, uint32_t
req_user_data_size)

Set the size of the user data buffer that will be allocated for each `doca_devemu_vfs_req` on behalf of the user. This buffer will be valid and used by the user upon receiving new `doca_devemu_vfs_req`. The buffer will become invalid after `doca_devemu_vfs_req` completion.

Parameters

vfs_dev

The DOCA Virtio FS device instance to modify. Must be idle.

req_user_data_size

Size, in bytes, of the user data buffer to be allocated on behalf of the user for each `doca_devemu_vfs_req`.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_dev' is NULL
- ▶ DOCA_ERROR_BAD_STATE - device is not idle

#define DOCA_VFS_TAG_SIZE 21

Size, in bytes, of the virtio FS tag in DOCA. According to the specification this is the name associated with the file system. The tag is encoded in UTF-8 and padded with NULL bytes if shorter than the available space. This field is not NULL terminated according to the Virtio specification. In DOCA, the tag encoding is shorter than in the Virtio specification and must be NULL terminated (only first 20 bytes are allowed to be encoded with non-NULL bytes).

2.10.2.1. DOCA Device Emulation - Virtio FS IO Context

DOCA Device Emulation - Virtio FS Devices

DOCA Virtio FS IO context

typedef

```
(*doca_devemu_vfs_io_event_vfs_notification_req_notice_handler_cb_t)
(doca_devemu_vfs_notification_req* req, void* req_user_data, union
doca_data_event_user_data)
```

Function to be executed on `vfs_notification_req_notice` event occurrence. The Ownership of the `doca_devemu_vfs_notification_req` and the `req_user_data` moves from `doca_devemu_vfs_io` ctx to the user.

```
typedef (*doca_devemu_vfs_io_event_vfs_req_notice_handler_cb_t)
(doca_devemu_vfs_req* req, void* req_user_data, union doca_data
event_user_data)
```

Function to be executed on `vfs_req_notice` event occurrence. The Ownership of the `doca_devemu_vfs_req` and the `req_user_data` moves from `doca_devemu_vfs_io` ctx to the user.

```
DOCA_EXPERIMENTAL doca_ctx *doca_devemu_vfs_io_as_ctx
(doca_devemu_vfs_io *io)
```

Convert DOCA Virtio FS device IO context instance into DOCA context.

Parameters

io

DOCA Virtio FS device IO context instance. This must remain valid until after the DOCA context is no longer required.

Returns

doca ctx upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_devemu_virtio_io
*doca_devemu_vfs_io_as_virtio_io (doca_devemu_vfs_io *io)
```

Convert DOCA Virtio FS device IO context instance into DOCA Virtio device IO context.

Parameters

io

DOCA Virtio FS device IO context instance. This must remain valid until after the DOCA Virtio device IO context is no longer required.

Returns

doca devemu virtio device io context upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_error_t doca_devemu_vfs_io_create
(doca_devemu_vfs_dev *vfs_dev, doca_pe *progress_engine,
doca_devemu_vfs_io **io)
```

Allocate Virtio FS device IO context for a DOCA Virtio FS device.

Parameters

vfs_dev

DOCA Virtio FS device.

progress_engine

The progress engine that will be used to progress the new context.

io

The created DOCA Virtio FS device IO context.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

Description

The responsibility of the Virtio FS IO context is to relay the requests arriving from the device driver towards the Virtio FS services and applications. Additionally, it is responsible for relaying the completions arriving from the Virtio FS services and applications towards the device driver. Each Virtio FS device IO context is associated with a single DOCA Virtio FS device.

DOCA_EXPERIMENTAL doca_error_t doca_devemu_vfs_io_destroy (doca_devemu_vfs_io *io)

Free a Virtio FS device IO context.

Parameters

io

The DOCA Virtio FS device IO context to release.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

DOCA_EXPERIMENTAL doca_error_t doca_devemu_vfs_io_event_vfs_notification_req_notice_register (doca_devemu_vfs_io *io, doca_devemu_vfs_io_event_vfs_notification_req_notice_handler_cb_t handler, doca_data user_data)

Register to Virtio FS notification_request notifications.

Parameters

io

The DOCA Virtio FS device IO context to be associated with the event. Must be idle.

handler

Method that is invoked once event is triggered.

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'io' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - IO is not idle

Description

Registration can be done only while IO ctx is idle. If called multiple times then only the last call will take effect.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_io_event_vfs_req_notice_register
(doca_devemu_vfs_io *io,
 doca_devemu_vfs_io_event_vfs_req_notice_handler_cb_t handler,
 doca_data user_data)
```

Register to Virtio FS request notifications.

Parameters

io

The DOCA Virtio FS device IO context to be associated with the event. Must be idle.

handler

Method that is invoked once event is triggered.

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'io' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - IO is not idle

Description

Registration can be done only while IO ctx is idle. If called multiple times then only the last call will take effect.

```
DOCA_EXPERIMENTAL void
doca_devemu_vfs_notification_req_complete
(doca_devemu_vfs_notification_req *req, uint32_t len)
```

Complete the Virtio FS notification_request. The Request ownership (including the associated dataout and req_user_data) moves from the user back to the associated IO context. The

associated IO context will complete the request towards the device driver according to the virtio fs specification.

Parameters

req

The Virtio FS notification_request to complete.

len

The number of bytes written into the device writable portion of the buffer described by the req.

```
DOCA_EXPERIMENTAL doca_buf
*doca_devemu_vfs_notification_req_get_dataout
(doca_devemu_vfs_notification_req *req)
```

Get the doca buffer associated with the dataout part of the Virtio FS notification request.

Parameters

req

The Virtio FS notification request to query.

Returns

The doca buffer representing the host memory for the device-writable part, virtio_fs_notify::(out_hdr + outarg), according to the virtio specification, associated to the notification request on success. NULL otherwise.

Description

This function should be issued during scheduling the request towards the execution context that will be writing to the doca buffer.

```
DOCA_EXPERIMENTAL void doca_devemu_vfs_req_complete
(doca_devemu_vfs_req *req, uint32_t len)
```

Complete the Virtio FS request. The Request ownership (including the associated datain, dataout and req_user_data) moves from the user back to the associated IO context. The associated IO context will complete the request towards the device driver according to the virtio fs specification.

Parameters

req

The Virtio FS request to complete.

len

The number of bytes written into the device writable portion of the buffer described by the req.

DOCA_EXPERIMENTAL doca_buf

***doca_devemu_vfs_req_get_datain (doca_devemu_vfs_req *req)**

Get the doca buffer associated with the datain part of the Virtio FS request.

Parameters

req

The Virtio FS request to query.

Returns

The doca buffer representing the host memory for the device-readable part, virtio_fs_req::(in + datain), according to the virtio specification, associated to the request on success. NULL otherwise.

Description

This function should be issued during scheduling the request towards the execution context that will be reading from the doca buffer.

DOCA_EXPERIMENTAL uint32_t

doca_devemu_vfs_req_get_datain_list_len (doca_devemu_vfs_req *req)

Get the number of elements in the original doca buffer linked list associated with the datain part of the Virtio FS request returned by doca_devemu_vfs_req_get_datain().

Parameters

req

The Virtio FS request to query. Must not be NULL.

Returns

Number of elements in the original datain doca buffer linked list. Valid only if the request is in the ownership of the user.

DOCA_EXPERIMENTAL doca_buf

***doca_devemu_vfs_req_get_dataout (doca_devemu_vfs_req *req)**

Get the doca buffer associated with the dataout part of the Virtio FS request.

Parameters

req

The Virtio FS request to query.

Returns

The doca buffer representing the host memory for the device-writable part, virtio_fs_req::(out + dataout), according to the virtio specification, associated to the request on success. NULL otherwise.

Description

This function should be issued during scheduling the request towards the execution context that will be writing to the doca buffer.

```
DOCA_EXPERIMENTAL uint32_t
doca_devemu_vfs_req_get_dataout_list_len (doca_devemu_vfs_req
*req)
```

Get the number of elements in the original doca buffer linked list associated with the dataout part of the Virtio FS request returned by doca_devemu_vfs_req_get_dataout().

Parameters

req

The Virtio FS request to query. Must not be NULL.

Returns

Number of elements in the original dataout doca buffer linked list. Valid only if the request is in the ownership of the user.

2.10.2.2. DOCA Device Emulation - Virtio FS Device Types

DOCA Device Emulation - Virtio FS Devices

DOCA Virtio FS type

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_find_default_vfs_type_by_dev (doca_dev *dev,
doca_devemu_vfs_type **vfs_type)
```

Find the default DOCA Virtio FS type associated with the device.

Parameters

dev

The doca dev associated with the default type.

vfs_type

Started DOCA Virtio FS default type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

DOCA_EXPERIMENTAL `doca_error_t`
`doca_devemu_vfs_is_default_vfs_type_supported (const`
`doca_devinfo *devinfo, uint8_t *supported)`

Check if the default DOCA Virtio FS type is supported by the device.

Parameters

devinfo

The device to query.

supported

1 if the default Virtio FS type is supported by the device, 0 otherwise.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'devinfo' or 'supported' are NULL.
- ▶ DOCA_ERROR_DRIVER - internal doca driver error

Description

Get `uint8_t` value defining if the device can be used to manage DOCA Virtio FS emulated devices associated with the default Virtio FS type.

DOCA_EXPERIMENTAL `doca_devemu_pci_type`
`*doca_devemu_vfs_type_as_pci_type (doca_devemu_vfs_type`
`*vfs_type)`

Convert DOCA Virtio FS type instance into DOCA PCI type.

Parameters

vfs_type

DOCA Virtio FS type instance. This must remain valid until after the DOCA PCI type is no longer required.

Returns

DOCA PCI type upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_devemu_virtio_type
*doca_devemu_vfs_type_as_virtio_type (doca_devemu_vfs_type
*vfs_type)
```

Convert DOCA Virtio FS type instance into DOCA Virtio type.

Parameters

vfs_type

DOCA Virtio FS type instance. This must remain valid until after the DOCA Virtio type is no longer required.

Returns

DOCA Virtio type upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_vfs_type_get_num_request_queues (const
doca_devemu_vfs_type *vfs_type, uint32_t *num_request_queues)
```

Get the value of the num_request_queues register.

Parameters

vfs_type

The DOCA Virtio FS type instance to query.

num_request_queues

The virtio_fs_config:num_request_queues register value according to virtio specification to be used, by default, by devices associated with the vfs_type.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'vfs_type' or 'num_request_queues' are NULL

2.10.3. DOCA Device Emulation - Virtio Devices

DOCA Device Emulation

DOCA library for emulated virtio devices logic

[DOCA Device Emulation - Virtio IO Context](#)

[DOCA Device Emulation - Virtio Device Types](#)

typedef

```
(*doca_devemu_virtio_dev_event_reset_handler_cb_t)
(doca_devemu_virtio_dev* virtio_dev, union doca_data
event_user_data)
```

Function to be executed on Virtio device reset. The event handler will enable users to quiesce, flush and reset the necessary resources associated with the emulated Virtio device. Upon event, all PCI I/O transactions to/from the host memory are disabled. Additionally, the user should flush all the outstanding resources associated with the emulated Virtio device, which were initially owned by the Virtio device and moved the the ownership of the user. After flushing all the outstanding resources, the user should call `doca_devemu_virtio_dev_reset_complete()`.

DOCA_EXPERIMENTAL doca_ctx

```
*doca_devemu_virtio_dev_as_ctx (doca_devemu_virtio_dev
*virtio_dev)
```

Convert DOCA Virtio device instance into DOCA context.

Parameters

virtio_dev

DOCA Virtio device instance. This must remain valid until after the DOCA context is no longer required.

Returns

doca ctx upon success, NULL otherwise.

DOCA_EXPERIMENTAL doca_devemu_pci_dev

```
*doca_devemu_virtio_dev_as_pci_dev
(doca_devemu_virtio_dev *virtio_dev)
```

Convert DOCA Virtio device instance into DOCA devemu PCI device.

Parameters

virtio_dev

DOCA Virtio device instance. This must remain valid until after the DOCA devemu PCI device is no longer required.

Returns

DOCA devemu pci device upon success, NULL otherwise.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_event_reset_register
(doca_devemu_virtio_dev *virtio_dev,
doca_devemu_virtio_dev_event_reset_handler_cb_t
handler, doca_data user_data)
```

Register to Virtio device reset event.

Parameters

virtio_dev

The DOCA Virtio dev context to be associated to the event. Must be idle.

handler

Method that is invoked once event is triggered.

user_data

User data that will be provided to the handler once invoked.

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - 'virtio_dev' or 'handler' are NULL
- ▶ DOCA_ERROR_BAD_STATE - virtio_dev context is not idle

Description

Registration can be done only if the Virtio device ctx is idle. If called multiple times then only the last call will take effect.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_config_generation
(const doca_devemu_virtio_dev *virtio_dev, uint8_t
*config_generation)
```

Get the Virtio config_generation register from common configuration structure according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

config_generation

The value of the config_generation register.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_config_msix_vector
(const doca_devemu_virtio_dev *virtio_dev, uint16_t
*config_msix_vector)
```

Get the Virtio `config_msix_vector` register according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

config_msix_vector

The value of the `config_msix_vector` register according to Virtio specification.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_device_features_63_0 (const
doca_devemu_virtio_dev *virtio_dev, uint64_t *features)
```

Get the Virtio `device_feature` bits (0-63) according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

features

The `device_feature` (bits 0-63) according to Virtio specification.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_device_status (const
doca_devemu_virtio_dev *virtio_dev, uint8_t *device_status)
```

Get the Virtio device_status register from common configuration structure according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

device_status

The value of the device_status register.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_driver_features_63_0 (const
doca_devemu_virtio_dev *virtio_dev, uint64_t *features)
```

Get the Virtio driver_feature bits (0-63) according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

features

The driver_feature (bits 0-63) according to Virtio specification.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_num_enabled_queues (const
doca_devemu_virtio_dev *virtio_dev, uint16_t *num_queues)
```

Get the number of enabled Virtio device queues by the driver. The driver enables a queue by setting the corresponding queue index to the queue_select register and setting the

queue_enable register to 1. The return value of num_queues is valid only if DRIVER_OK status bit was set by the driver.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

num_queues

The number of enable Virtio queues for the virtio device.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_num_queues (const
doca_devemu_virtio_dev *virtio_dev, uint16_t *num_queues)
```

Get the Virtio device num_queues register from common configuration structure according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

num_queues

The value of the num_queues register.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_num_required_running_virtio_io_ctxs
(const doca_devemu_virtio_dev *virtio_dev, uint32_t
*num_virtio_io)
```

Get the number of required running Virtio io context's to be bounded to the Virtio device context. The Virtio device context will not move to a "running" state before having this amount of running Virtio IO context's bounded to it.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

num_virtio_io

The number of required running Virtio IO ctx's to be bounded to the device.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_get_queue_size (const
doca_devemu_virtio_dev *virtio_dev, uint16_t *queue_size)
```

Get the Virtio max queue size for all Virtio queues.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

queue_size

The maximal queue size for all Virtio queues.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_reset_complete
(doca_devemu_virtio_dev *virtio_dev)
```

Complete the Virtio device reset handling. Prior to calling this function, the user must ensure that all the resources associated with the Virtio device are flushed back to the ownership of the device.

Parameters

virtio_dev

DOCA Virtio device instance.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_set_device_features_63_0
(doca_devemu_virtio_dev *virtio_dev, uint64_t features)
```

Set the Virtio `device_feature` bits (0-63) according to Virtio specification.

Parameters

virtio_dev

The DOCA Virtio device instance to modify.

features

The device_feature (bits 0-63) according to Virtio specification.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_set_num_queues
(doca_devemu_virtio_dev *virtio_dev, uint16_t num_queues)**

Set the Virtio device num_queues register in common configuration structure according to Virtio specification.

Parameters**virtio_dev**

The DOCA Virtio device instance to modify.

num_queues

The device common num_queues register.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

**DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_set_num_required_running_virtio_io_ctxs
(doca_devemu_virtio_dev *virtio_dev, uint32_t
num_virtio_io)**

Set the number of required running Virtio IO context's to be bounded to the Virtio device context. The Virtio device context will not move to a "running" state before having this amount of running Virtio IO context's bounded to it.

Parameters**virtio_dev**

The DOCA Virtio device instance to modify.

num_virtio_io

The number of required running Virtio IO ctx's to be bounded.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_dev_set_queue_size
(doca_devemu_virtio_dev *virtio_dev, uint16_t queue_size)
```

Set the Virtio max queue size for all Virtio queues.

Parameters

virtio_dev

The DOCA Virtio device instance to query.

queue_size

The maximal queue size for all Virtio queues.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

2.10.3.1. DOCA Device Emulation - Virtio IO Context

DOCA Device Emulation - Virtio Devices

DOCA VIRTIO IO context

```
DOCA_EXPERIMENTAL doca_ctx *doca_devemu_virtio_io_as_ctx
(doca_devemu_virtio_io *io)
```

Convert DOCA Virtio device IO context instance into doca context.

Parameters

io

DOCA Virtio device IO context instance. This must remain valid until after the context is no longer required.

Returns

doca ctx upon success, NULL otherwise.

2.10.3.2. DOCA Device Emulation - Virtio Device Types

DOCA Device Emulation - Virtio Devices

DOCA Virtio type

`DOCA_EXPERIMENTAL doca_error_t
 doca_devemu_virtio_cap_default_type_get_configurable_device_features_63_0
 (const doca_devinfo *devinfo, uint64_t *features)`

Get a bitmap of configurable device feature bits (0-63) for Virtio devices associated with a default virtio type.

Parameters

devinfo

The device to query.

features

Bitmap of configurable device feature bits for all Virtio device associated with the default Virtio type. Feature bit indices follow the Virtio specification.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see `doca_error_t`.

`DOCA_EXPERIMENTAL doca_devemu_pci_type
 *doca_devemu_virtio_type_as_pci_type (doca_devemu_virtio_type
 *virtio_type)`

Convert DOCA Virtio type instance into DOCA PCI type.

Parameters

virtio_type

DOCA Virtio type instance. This must remain valid until after the DOCA PCI type is no longer required.

Returns

DOCA PCI type upon success, NULL otherwise.

`DOCA_EXPERIMENTAL doca_error_t
 doca_devemu_virtio_type_get_config_generation (const
 doca_devemu_virtio_type *virtio_type, uint8_t *config_generation)`

Get the initial Virtio device `config_generation` register according to Virtio specification configured for this type.

Parameters

virtio_type

The DOCA Virtio type instance to query.

config_generation

The initial value of the config_generation register according to Virtio specification for devices associated with this type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_type_get_device_features_63_0 (const doca_devemu_virtio_type *virtio_type, uint64_t *features)

Get the Virtio device_feature bits (0-63) according to Virtio specification configured for this type.

Parameters**virtio_type**

The DOCA Virtio type instance to query.

features

The device_feature (bits 0-63) according to Virtio specification for devices associated with this type.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_type_get_num_queues (const doca_devemu_virtio_type *virtio_type, uint16_t *num_queues)

Get the value of the num_queues register.

Parameters**virtio_type**

The DOCA Virtio type instance to query.

num_queues

The default value to be used by devices associated with the virtio_type for virtio_common_config:num_queues register according to Virtio specification, if not set otherwise.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

```
DOCA_EXPERIMENTAL doca_error_t
doca_devemu_virtio_type_get_queue_size (const
doca_devemu_virtio_type *virtio_type, uint16_t *queue_size)
```

Get the value of the queue_size register.

Parameters

virtio_type

The DOCA Virtio type instance to query.

queue_size

The default value to be used by devices associated with the virtio_type for virtio_common_config:queue_size register according to Virtio specification, if not set otherwise.

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure. see doca_error_t.

2.11. DOCA DMA Engine

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

```
typedef (*doca_dma_task_memcpy_completion_cb_t)
(doca_dma_task_memcpy* task, union doca_data
task_user_data, union doca_data ctx_user_data)
```

Function to execute on DMA memcpy task completion.

This function is called by doca_pe_progress() when related task identified as completed successfully. When this function called the ownership of the task object passed from DOCA back to user. Inside this callback user may decide on the task object:

- ▶ re-submit task with doca_task_submit(); task object ownership passed to DOCA
- ▶ release task with doca_task_free(); task object ownership passed to DOCA
- ▶ keep the task object for future re-use; user keeps the ownership on the task object Inside this callback the user shouldn't call doca_pe_progress(). Please see doca_pe_progress for details.

Any failure/error inside this function should be handled internally or deferred; due to the mode of nested in doca_pe_progress() execution this callback doesn't return error.

NOTE: this callback type utilized for both successful & failed task completions.

```
DOCA_STABLE 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 generalized context for use with doca core objects.

```
DOCA_STABLE doca_error_t
doca_dma_cap_get_max_num_tasks (doca_dma
*dma, uint32_t *max_num_tasks)
```

Parameters

dma

The dma context

max_num_tasks

Max number of memcopy tasks

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 log number of tasks

This method retrieves the maximum number of tasks for a device.

```
DOCA_STABLE doca_error_t
doca_dma_cap_task_memcpy_get_max_buf_list_len
(const doca_devinfo *devinfo, uint32_t
*max_buf_list_len)
```

Parameters

devinfo

The DOCA device information.

max_buf_list_len

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 memcpy task.

```
DOCA_STABLE doca_error_t
doca_dma_cap_task_memcpy_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 memcpy task.

```
DOCA_STABLE doca_error_t
doca_dma_cap_task_memcpy_is_supported (const
doca_devinfo *devinfo)
```

Parameters

devinfo

The DOCA device information

Returns

DOCA_SUCCESS - in case device supports memcpy. Error code - in case of failure:

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
- ▶ DOCA_ERROR_DRIVER - failed to query the device for its capabilities.
- ▶ DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support memcpy.

Description

Check if given device is capable of executing DMA memcpy task.

```
DOCA_STABLE doca_error_t doca_dma_create
(doca_dev *dev, doca_dma **dma)
```

Parameters

dev

The device to attach to the DMA context

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 initialize a mutex.

Description

Create a DOCA DMA instance.

```
DOCA_STABLE 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_EXPERIMENTAL doca_error_t
doca_dma_get_gpu_handle (doca_dma *dma,
doca_gpu_dma **gpu_dma)
```

Retrieve the handle in the GPU memory space of a doca_dma.

Parameters

dma

doca_dma context to get the GPU handle from.

gpu_dma

A pointer to the handle in the gpu memory space.

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 called before calling ctx_start(), or if not assigned to gpu datapath.

```
DOCA_STABLE doca_error_t
doca_dma_task_memcpy_alloc_init (doca_dma
*dma, const doca_buf *src, doca_buf *dst, doca_data
user_data, doca_dma_task_memcpy **task)
```

This method allocates and initializes a DMA memcpy task.

Parameters

dma

The DMA to allocate the task for

src

source buffer

dst

destination buffer

user_data

doca_data to attach to the task

task

memcpy task to allocate

Returns

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

- ▶ DOCA_ERROR_NO_MEMEORY - No more tasks to allocate

```
DOCA_STABLE doca_task
*doca_dma_task_memcpy_as_task
(doca_dma_task_memcpy *task)
```

This method converts a memcpy task to doca_task.

Parameters

task

doca_dma_task_memcpy task

Returns

doca_task

DOCA_STABLE doca_buf

```
*doca_dma_task_memcpy_get_dst (const  
doca_dma_task_memcpy *task)
```

This method gets destination buffer from memcpy task.

Parameters

task

The task to get

Returns

destination buffer

DOCA_STABLE doca_buf

```
*doca_dma_task_memcpy_get_src (const  
doca_dma_task_memcpy *task)
```

This method gets source buffer from memcpy task.

Parameters

task

The task to get

Returns

source buffer

DOCA_STABLE doca_error_t

```
doca_dma_task_memcpy_set_conf (doca_dma  
*dma, doca_dma_task_memcpy_completion_cb_t  
task_completion_cb,  
doca_dma_task_memcpy_completion_cb_t  
task_error_cb, uint32_t num_memcpy_tasks)
```

This method sets the DMA memcpy tasks configuration.

Parameters

dma

The DMA context to config

task_completion_cb

Task completion callback

task_error_cb

Task error callback

num_memcpy_tasks

Number of memcpy tasks that the DMA can allocate

Returns

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

- ▶ DOCA_ERROR_INVALID_VALUE - doca_pe_dma argument is a NULL pointer.
- ▶ DOCA_ERROR_NOT_SUPPORTED - context is in work queue mode

DOCA_STABLE void

```
doca_dma_task_memcpy_set_dst
(doca_dma_task_memcpy *task, doca_buf *dst)
```

This method sets destination buffer to memcpy task.

Parameters**task**

The task to set

dst

Destination buffer

DOCA_STABLE void

```
doca_dma_task_memcpy_set_src
(doca_dma_task_memcpy *task, const doca_buf *src)
```

This method sets source buffer to memcpy task.

Parameters**task**

The task to set

src

Source buffer

2.12. DOCA DPA Host

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

enum doca_dpa_dev_log_level_t

DOCA DPA device log levels, sorted by verbosity from high to low.

Values

DOCA_DPA_DEV_LOG_LEVEL_DISABLE = 10

Disable log messages

DOCA_DPA_DEV_LOG_LEVEL_CRIT = 20

Critical log level

DOCA_DPA_DEV_LOG_LEVEL_ERROR = 30

Error log level

DOCA_DPA_DEV_LOG_LEVEL_WARNING = 40

Warning log level

DOCA_DPA_DEV_LOG_LEVEL_INFO = 50

Info log level

DOCA_DPA_DEV_LOG_LEVEL_DEBUG = 60

Debug log level

typedef uint64_t doca_dpa_dev_async_ops_t

DPA asynchronous ops handle type definition.

typedef uint64_t doca_dpa_dev_completion_t

DPA completion handle type definition.

typedef uint64_t doca_dpa_dev_hash_table_t

DPA hash table handle type definition.

typedef uint64_t

doca_dpa_dev_notification_completion_t

DPA notification completion handle type definition.

typedef uint64_t doca_dpa_dev_t

DPA context handle type definition.

typedef uint64_t doca_dpa_dev_uintptr_t

DPA pointer type definition.

typedef void (doca_dpa_func_t)

Generic function pointer type.

Kernel launches are made using a host function pointer that represents the device function. The host function stub is provided by the associated DPA compiler. The C language does not define conversion of a function pointer to an object pointer (such as void*). Programmers can use this generic function pointer type to typecast to and adhere to strict ISO C language requirements

DOCA_EXPERIMENTAL doca_error_t doca_dpa_app_get_name (doca_dpa_app *app, char *app_name, uint32_t *app_name_len)

Get DPA application name.

Parameters

app

- DPA application generated by DPACC

app_name

- application name

app_name_len

- app_name length. Output is actual number of bytes written

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input, or the buffer received is of insufficient length
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call

Description

The name of a DPA application is assigned using DPACC during the build phase. Once an application has been formed, its name is embedded within it. This function allows DOCA DPA's host application to retrieve the name that was previously assigned.

The app_name buffer is allocated by the caller along with setting app_name_len indicating the length that was allocated. Upon return the app_name_len field is set to the actual length of the app_name

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_async_ops_attach (doca_dpa_async_ops
*async_ops, doca_dpa_completion *dpa_comp)
```

Attach DPA asynchronous ops context to DPA completion context.

Parameters

async_ops

- DPA asynchronous ops

dpa_comp

- DPA completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA asynchronous ops is already started

Description

This function attaches DPA asynchronous ops context to DPA completion context. Once a context is attached and both contexts are started, asynchronous operations completion will be raised on the DPA completion (in case the user asks for a completion when issuing the operation). This function must be called before DPA asynchronous ops is started

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_async_ops_create (doca_dpa *dpa,
unsigned int queue_size, uint64_t user_data,
doca_dpa_async_ops **async_ops)
```

Create DPA asynchronous ops context.

Parameters

dpa

- DPA context

queue_size

- DPA asynchronous ops queue size

user_data

- DPA asynchronous ops user data

async_ops

- created DPA asynchronous ops

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NO_MEMORY - failure in internal memory allocation

Description

This function creates a DPA asynchronous ops context. This allows the DPA thread to issue asynchronous operations, like DMA or other operations. User can provide DPA asynchronous ops context `user_data`, and retrieve this metadata in device using `doca_dpa_dev_get_completion_user_data()` API

**Note:**

Queue size will be rounded to the next power of 2

DOCA_EXPERIMENTAL doca_error_t doca_dpa_async_ops_destroy (doca_dpa_async_ops *async_ops)

Destroy DPA asynchronous ops context.

Parameters**async_ops**

- Previously created DPA asynchronous ops

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA asynchronous ops context created by [doca_dpa_async_ops_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_async_ops_get_dpa_handle  
(doca_dpa_async_ops *async_ops,  
doca_dpa_dev_async_ops_t *handle)
```

Get DPA asynchronous ops context handle.

Parameters

async_ops

- DPA asynchronous ops

handle

- DPA asynchronous ops handle

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA asynchronous ops is not started

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_async_ops_get_queue_size  
(doca_dpa_async_ops *async_ops, unsigned int  
*queue_size)
```

Get DPA asynchronous ops queue size.

Parameters

async_ops

- DPA asynchronous ops

queue_size

- DPA asynchronous ops queue size

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_async_ops_get_user_data
(doca_dpa_async_ops *async_ops, uint64_t
*user_data)
```

Get DPA asynchronous ops context user data.

Parameters

async_ops

- DPA asynchronous ops

user_data

- DPA asynchronous ops user data

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_async_ops_start (doca_dpa_async_ops
*async_ops)
```

Start DPA asynchronous ops context.

Parameters

async_ops

- DPA asynchronous ops

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_async_ops_stop (doca_dpa_async_ops  
*async_ops)
```

Stop DPA asynchronous ops context.

Parameters

async_ops

- DPA asynchronous ops

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_cap_is_supported (const doca_devinfo  
*devinfo)
```

Get whether the DOCA device supports DPA.

Parameters

devinfo

- the device to query

Returns

- ▶ DOCA_SUCCESS - in case of the DOCA device queried has DPA support
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NOT_SUPPORTED - the device queried does not support DPA

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_completion_create (doca_dpa *dpa,
unsigned int queue_size, doca_dpa_completion
**dpa_comp)
```

Create DPA completion context.

Parameters

dpa

- DPA context

queue_size

- DPA completion queue size

dpa_comp

- created DPA completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NO_MEMORY - failure in internal memory allocation

Description

This function creates a DPA completion context. The user is responsible for creating and managing the context. The completion context can raise activation if it is attached to a DPA thread. The user can also decide to progress the context via polling it manually



Note:

Queue size will be rounded to the next power of 2

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_completion_destroy (doca_dpa_completion
*dpa_comp)
```

Destroy DPA completion context.

Parameters

dpa_comp

- Previously created DPA completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA completion context created by [doca_dpa_completion_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_completion_get_dpa_handle
(doca_dpa_completion *dpa_comp,
doca_dpa_dev_completion_t *handle)
```

Get DPA completion context handle.

Parameters

dpa_comp

- DPA completion

handle

- DPA completion handle

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA completion is not started

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_completion_get_queue_size
(doca_dpa_completion *dpa_comp, unsigned int
*queue_size)
```

Get DPA completion context queue size.

Parameters

dpa_comp

- DPA completion

queue_size

- DPA completion queue size

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

DOCA_EXPERIMENTAL doca_error_t
 doca_dpa_completion_get_thread
 (doca_dpa_completion *dpa_comp, doca_dpa_thread
 **dpa_thread)

Get DPA completion context attached thread.

Parameters

dpa_comp

- DPA completion

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

DOCA_EXPERIMENTAL doca_error_t
 doca_dpa_completion_set_thread
 (doca_dpa_completion *dpa_comp, doca_dpa_thread
 *dpa_thread)

Set DPA completion context thread.

Parameters

dpa_comp

- DPA completion

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA completion is already started

Description

This function attaches DPA thread to DPA completion context. Attaching to a DPA thread is required if the user wants activation of the thread when a completion is raised on the completion context. This function must be called before DPA completion is started

DOCA_EXPERIMENTAL doca_error_t doca_dpa_completion_start (doca_dpa_completion *dpa_comp)

Start DPA completion context.

Parameters

dpa_comp

- DPA completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

DOCA_EXPERIMENTAL doca_error_t doca_dpa_completion_stop (doca_dpa_completion *dpa_comp)

Stop DPA completion context.

Parameters

dpa_comp

- DPA completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call

- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_create (doca_dev *dev, doca_dpa **dpa)
```

Create a DOCA DPA Context.

Parameters

dev

- DOCA device

dpa

- created context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_NOT_SUPPORTED - the device does not support DPA
- ▶ DOCA_ERROR_NO_MEMORY - in case of failure in internal memory allocation

Description

This function creates a DOCA DPA context given a DOCA device. The context represents a program on the DPA that is referenced by the host process that called the context creation API

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_d2h_buf_memcpy (doca_dpa *dpa,
doca_buf *buf, doca_dpa_dev_uintptr_t src_ptr, size_t
size)
```

Copy from DPA Heap to DOCA Buf.

Parameters

dpa

- DPA context

buf

- destination DOCA Buf

src_ptr

- DPA device heap source pointer

size

- size of data to copy

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function copies data from the DPA heap to start of DOCA Buf. This is a blocking call. When the call returns, the memory on the DOCA Buf is set to the values supplied in the DPA heap pointer

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_d2h_memcpy (doca_dpa *dpa, void
***dst_ptr, doca_dpa_dev_uintptr_t src_ptr, size_t size)**

Copy from DPA Heap to host memory.

Parameters**dpa**

- DPA context

dst_ptr

- host destination buffer address

src_ptr

- DPA device heap source pointer

size

- size of data to copy

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function copies data from the DPA heap to Host memory. This is a blocking call. When the call returns, the memory on the Host buffer is set to the values supplied in the DPA heap pointer

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_destroy (doca_dpa *dpa)
```

Destroy a DOCA DPA context.

Parameters

dpa

- Previously created DPA context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_IN_USE - in case the DPA context is still used by another DOCA context

Description

This function destroys DPA context created by [doca_dpa_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_device_extend (doca_dpa *dpa, doca_dev
*other_dev, doca_dpa **extended_dpa)
```

Create an extended DPA context.

Parameters

dpa

- Base DPA context

other_dev

- DOCA device to be extend to.

extended_dpa

- Created extended DPA context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE
- ▶ DOCA_ERROR_NO_MEMORY - in case of failure in internal memory allocation
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call

Description

This function extends a base DPA context (was created on PF DOCA device) to other DOCA device (VF/SF device) and creates an extended DPA context. This to allow creation of DPA resources such as RDMA/DPA completion/DPA Async ops... contexts on the other device. The extended DPA context can be used later on for all DOCA DPA APIs such as creating DPA memory/DPA completion context/... Please note:

- ▶ The returned DPA context will be already started.
- ▶ After creating the extended DPA context, user must destroy it before destroying the base DPA context.

DOCA_EXPERIMENTAL doca_error_t doca_dpa_eu_affinity_clear (doca_dpa_eu_affinity *affinity)

Clear DPA EU affinity.

Parameters

affinity

- DPA EU affinity

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

This function clears EU ID in the given DPA EU affinity

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_eu_affinity_create (doca_dpa *dpa,
doca_dpa_eu_affinity **affinity)

Create DPA EU affinity.

Parameters

dpa

- DPA context

affinity

- created DPA EU affinity

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

This function creates DPA EU affinity. DPA EU affinity can be set for a DPA thread using [doca_dpa_thread_set_affinity\(\)](#) to specify EU ID which DPA thread will use on DPA. Look at [sched_getaffinity\(3\)](#) for corresponding CPU use case. We are replicating it here for DPA

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_eu_affinity_destroy (doca_dpa_eu_affinity
***affinity)**

Destroy DPA EU affinity.

Parameters

affinity

- previously created DPA EU affinity

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA EU affinity created by [doca_dpa_eu_affinity_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_eu_affinity_get (doca_dpa_eu_affinity  
*affinity, unsigned int *eu_id)
```

Get EU ID from DPA EU affinity.

Parameters

affinity

- DPA EU affinity

eu_id

- EU ID

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

This function gets EU ID from the given DPA EU affinity

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_eu_affinity_set (doca_dpa_eu_affinity  
*affinity, unsigned int eu_id)
```

Set EU ID in DPA EU affinity.

Parameters

affinity

- DPA EU affinity

eu_id

- EU ID

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

This function sets EU ID in the given DPA EU affinity

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_app (doca_dpa *dpa, doca_dpa_app
**app)
```

Get program app that was set for DPA context.

Parameters

dpa

- DPA context

app

- DPA application set for DPA context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_core_num (doca_dpa *dpa, unsigned
int *num_cores)
```

Retrieve the number of available DPA cores.

Parameters

dpa

- DPA context

num_cores

- number of DPA cores

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

Use this info to select EU ID for DPA EU affinity

DOCA_EXPERIMENTAL `doca_error_t`
`doca_dpa_get_dpa_handle (doca_dpa *dpa,`
`doca_dpa_dev_t *handle)`

Get DPA context handle.

Parameters

dpa

- DPA context

handle

- DPA context handle

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA completion is not started

DOCA_EXPERIMENTAL `doca_error_t`
`doca_dpa_get_kernel_max_run_time (const`
`doca_dpa *dpa, unsigned long long *value)`

Get maximum allowable time in seconds that a kernel may remain scheduled on the DPA. A kernel that remains scheduled beyond this limit may be terminated by the runtime and cause fatal behavior.

Parameters

dpa

- DPA context

value

- maximum allowed time in seconds for a kernel to remain scheduled

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid NULL input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_log_level (doca_dpa *dpa,
doca_dpa_dev_log_level_t *log_level)
```

Get device logs verbosity level that was set for DPA context.

Parameters

dpa

- DPA context

log_level

- verbosity level for device logs

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid null input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_max_threads_per_kernel (const
doca_dpa *dpa, unsigned int *value)
```

Get maximum number of DPA threads to run a single kernel launch operation.

Parameters

dpa

- DPA context

value

- number of maximum threads to run a kernel

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid NULL input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_num_eus_per_core (doca_dpa *dpa,
unsigned int *eus_per_core)
```

Retrieve the number of EUs available per DPA core.

Parameters

dpa

- DPA context

eus_per_core

- number of EUs per DPA core

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

Use this info to select EU ID for DPA EU affinity

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_get_total_num_eus_available (doca_dpa
*dpa, unsigned int *total_num_eus)
```

Retrieve the total number of EUs available to the application.

Parameters

dpa

- DPA context

total_num_eus

- number of total available EUs

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

Description

Use this info to select EU ID for DPA EU affinity

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_h2d_buf_memcpy (doca_dpa *dpa,
doca_dpa_dev_uintptr_t dst_ptr, doca_buf *buf, size_t
size)
```

Copy from DOCA Buf to DPA Heap.

Parameters

dpa

- DPA context

dst_ptr

- DPA device heap destination pointer

buf

- source DOCA Buf

size

- size of data to copy

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function copies data from start of DOCA Buf to the DPA heap. This is a blocking call. When the call returns, the memory on the DPA is set to the values supplied in the DOCA Buf

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_h2d_memcpy (doca_dpa *dpa,
doca_dpa_dev_uintptr_t dst_ptr, void *src_ptr, size_t
size)
```

Copy from host memory to DPA Heap.

Parameters

dpa

- DPA context

dst_ptr

- DPA device heap destination pointer

src_ptr

- host source buffer address

size

- size of data to copy

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function copies data from Host memory to the DPA heap. This is a blocking call. When the call returns, the memory on the DPA is set to the values supplied in the Host buffer

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_hash_table_create (doca_dpa *dpa,
unsigned int num_entries, doca_dpa_hash_table **ht)
```

Create a hash table on DPA.

Parameters**dpa**

- DPA context

num_entries

- number of entries in the hash table

ht

- Created DPA hash table

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function creates a hash table on DPA. Once it is created, user can retrieve its DPA handle and use it for add, remove and find operations in DPA kernels



Note:

Table size will be rounded to the next power of 2

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_hash_table_destroy (doca_dpa_hash_table
*ht)
```

Destroy DPA hash table.

Parameters

ht

- Previously created DPA hash table

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

Description

This function destroys DPA hash table created by [doca_dpa_hash_table_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_hash_table_get_dpa_handle
(doca_dpa_hash_table *ht,
doca_dpa_dev_hash_table_t *ht_handle)
```

Get DPA hash table handle.

Parameters

ht

- DPA hash table

ht_handle

- DPA hash table handle

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_kernel_launch_update_add (doca_dpa
*dpa, doca_sync_event *wait_event, uint64_t
wait_threshold, doca_sync_event *comp_event,
uint64_t comp_count, unsigned int num_threads,
doca_dpa_func_t *func, ...)
```

Submit a kernel to DPA.

Parameters**dpa**

- previously created DPA context

wait_event

- event to wait on before executing the kernel (optional)

wait_threshold

- wait event count threshold to wait for before executing. Valid values [0-254]

comp_event

- event to signal after kernel execution is complete (optional)

comp_count

- completion count to add for completion event when func is complete

num_threads

- number of threads to use. This number must be equal or lower than the maximum allowed (see `doca_dpa_get_max_threads_per_kernel`)

func

- host function pointer representing DPA kernel

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function submits a kernel for launch on the specified `dpa` context. The kernel starts execution when its wait event value is greater than or equal to specified threshold. The value specified in `comp_count` is added to the `comp_event` when the kernel finishes execution.

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_kernel_launch_update_set (doca_dpa
*dpa, doca_sync_event *wait_event, uint64_t
wait_threshold, doca_sync_event *comp_event,
uint64_t comp_count, unsigned int num_threads,
doca_dpa_func_t *func, ...)
```

Submit a kernel to DPA that sets completion event.

Parameters

dpa

- previously created DPA context

wait_event

- event to wait on before executing the kernel (optional)

wait_threshold

- wait event count threshold to wait for before executing. Valid values [0-254]

comp_event

- event to signal after kernel execution is complete (optional)

comp_count

- completion count to set for completion event when func is complete

num_threads

- number of threads to use. This number must be equal or lower than the maximum allowed (see `doca_dpa_get_max_threads_per_kernel`)

func

- host function pointer representing DPA kernel

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function submits a kernel for launch on the specified `dpa` context. The kernel starts execution when its wait event value is greater than or equal to specified threshold. The completion event is set to value specified in `comp_count` when the kernel finishes execution.

The function to be launched `func` is a host function pointer corresponding to the DPA device function. For example, if the device function is declared as: `__dpa_global__ hello(int arg1)`, then the user is expected to declare the function in the Host application as `extern doca_dpa_func_t hello;`. After the application is linked and loaded using the compiler, a function pointer `hello` can be used in as the `func` argument. The arguments to the function `hello` can be passed inline in the call as var args. For example, to call `hello` on the device using `4` threads with argument `5`, the invocation looks like: `doca_dpa_kernel_launch_update_set(..., 4, hello, 5);`

DOCA_EXPERIMENTAL doca_error_t doca_dpa_log_file_get_path (doca_dpa *dpa, char *file_path, uint32_t *file_path_len)

Get log file path to write device logs into.

Parameters

dpa

- DPA context

file_path

- pathname to the log file to write device logs into

file_path_len

- file_path length. Output is actual number of bytes written

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid null input
- ▶ DOCA_ERROR_NO_MEMORY - memory allocation error

Description

The file_path buffer is allocated by the caller along with setting file_path_len indicating the length that was allocated. Upon return the file_path_len field is set to the actual length of the file_path

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_log_file_set_path (doca_dpa *dpa, const
char *file_path)
```

Set log file path to write device logs into.

Parameters

dpa

- DPA context

file_path

- pathname to the log file to write device logs into

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid null input
- ▶ DOCA_ERROR_BAD_STATE - DPA context is already started
- ▶ DOCA_ERROR_NO_MEMORY - memory allocation error
- ▶ DOCA_ERROR_OPERATING_SYSTEM - error occurred in opening the file

Description

Must be set before calling [doca_dpa_start\(\)](#)



Note:

if not set then stdout will be used by default

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_mem_alloc (doca_dpa *dpa, size_t size,
doca_dpa_dev_uintptr_t *dev_ptr)
```

Allocate DPA heap memory.

Parameters

dpa

- DPA context

size

- requested size of allocation

dev_ptr

- pointer to the allocated memory on the DPA device

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function allocates memory of `size` bytes on the DPA process heap. The memory is aligned for any language supported data type. The memory is not zeroed on allocation. The allocated memory is returned in `dev_ptr` when successful. When memory allocation fails, `dev_ptr` is set to 0x0 (NULL)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_mem_free (doca_dpa *dpa,
doca_dpa_dev_uintptr_t dev_ptr)
```

Free the previously allocated DPA memory.

Parameters**dpa**

- DPA context

dev_ptr

- pointer to the memory that was previously allocated on the DPA device

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function frees the allocated memory allocated on the DPA heap. Users are expected to ensure that kernels on the DPA are no longer accessing the memory using established synchronization mechanisms (see events)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_memset (doca_dpa *dpa,
doca_dpa_dev_uintptr_t dev_ptr, int value, size_t size)
```

Set DPA Heap memory to a value.

Parameters

dpa

- DPA context

dev_ptr

- DPA device heap pointer

value

- value to set

size

- size of device buffer

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - DPA context is not started

Description

This function sets DPA heap memory to a supplied value. This is a blocking call. When the call returns, the memory on the DPA is set to the value supplied

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_create
(doca_dpa *dpa, doca_dpa_thread *dpa_thread,
doca_dpa_notification_completion **notify_comp)
```

Create DPA notification completion context.

Parameters

dpa

- DPA context

dpa_thread

- attached DPA thread

notify_comp

- created DPA notification completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NO_MEMORY - failure in internal memory allocation

Description

This function creates a DPA notification completion context. This context is used to activate the attached DPA thread in device using `doca_dpa_dev_thread_notify()` API. The thread activation is done without receiving a completion on the thread's attached completion context. Therefore it is expected that the user of this method of thread activation will pass the message in another fashion – such as shared memory

**DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_destroy
(doca_dpa_notification_completion *notify_comp)**

Destroy DPA notification completion context.

Parameters**notify_comp**

- Previously created DPA notification completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA notification completion context created by [doca_dpa_notification_completion_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_get_dpa_handle
(doca_dpa_notification_completion *notify_comp,
doca_dpa_dev_notification_completion_t *handle)
```

Get DPA notification completion context handle.

Parameters

notify_comp

- DPA notification completion

handle

- DPA notification completion handle

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA notification completion is not started

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_get_thread
(doca_dpa_notification_completion *notify_comp,
doca_dpa_thread **dpa_thread)
```

Get DPA notification completion context attached thread.

Parameters

notify_comp

- DPA notification completion

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_start
(doca_dpa_notification_completion *notify_comp)

Start DPA notification completion context.

Parameters

notify_comp

- DPA notification completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_notification_completion_stop
(doca_dpa_notification_completion *notify_comp)

Stop DPA notification completion context.

Parameters

notify_comp

- DPA notification completion

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_peek_at_last_error (const doca_dpa *dpa)

Return the last error generated on the DPA. Check if an error occurred on the device side runtime. This call does not reset the error state. If an error occurred, the DPA context enters a fatal state and must be destroyed by the user. In the case of a fatal error core dump and crash data will be written to the file path /tmp/doca_dpa_fatal or to the file path set by the API

`doca_dpa_log_file_set_path()`, with the suffixes `.PID.core` and `.PID.crash` respectively, where PID is the process id.

Parameters

dpa

- DPA context

Returns

- ▶ `DOCA_SUCCESS` - in case of success
- ▶ `DOCA_ERROR_INVALID_VALUE` - received invalid NULL input
- ▶ `DOCA_ERROR_BAD_STATE` - received error on device side

`DOCA_EXPERIMENTAL doca_error_t doca_dpa_rpc`
`(doca_dpa *dpa, doca_dpa_func_t *func, uint64_t`
`*retval, ...)`

RPC to run DPA kernel.

Parameters

dpa

- DPA context

func

- Host function pointer representing DPA kernel to run

retval

- A pointer to the DPA kernel return value

Returns

- ▶ `DOCA_SUCCESS` - in case of success
- ▶ `DOCA_ERROR_INVALID_VALUE` - received invalid input
- ▶ `DOCA_ERROR_DRIVER` - error in a DOCA driver call
- ▶ `DOCA_ERROR_BAD_STATE` - attached DPA context is not started

Description

This function executes the supplied `func` with its argument on the specified `dpa` context. This is a blocking API.

`func` is a host function pointer corresponding to the DPA device function. Please note that DPA device `func` must be annotated with `__dpa_rpc__` annotation, such as `__dpa_rpc__ uint64_t hello(int arg1)`. Also the user is expected to declare the function in the Host application as `extern doca_dpa_func_t hello;`

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_set_app (doca_dpa *dpa, doca_dpa_app
*app)
```

Set program app for DPA context.

Parameters

dpa

- DPA context

app

- DPA application generated by DPACC

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - if DPA context is already started

Description

The program app represents a program on the DPA that is referenced by the host process that called the context creation API. Must be set before calling [doca_dpa_start\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_set_log_level (doca_dpa *dpa,
doca_dpa_dev_log_level_t log_level)
```

Set device logs verbosity level.

Parameters

dpa

- DPA context

log_level

- verbosity level for device logs

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA context is already started

Description

Log prints from the device will be printed to host with verbosity that is equal or lower than the set log level. Must be set before calling [doca_dpa_start\(\)](#)



Note:

Default value of DPA log level is DOCA_DPA_DEV_LOG_LEVEL_INFO

DOCA_EXPERIMENTAL doca_error_t doca_dpa_start (doca_dpa *dpa)

Start a DPA context.

Parameters

dpa

- DPA context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_NO_MEMORY - in case of failure in internal memory allocation

DOCA_EXPERIMENTAL doca_error_t doca_dpa_stop (doca_dpa *dpa)

Stop a DPA context.

Parameters

dpa

- DPA context

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - in case of error in a DOCA driver call
- ▶ DOCA_ERROR_IN_USE - in case of attached thread/completion/extended DPA context that is not destroyed

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_create (doca_dpa *dpa,
doca_dpa_thread **dpa_thread)
```

Create DPA thread.

Parameters

dpa

- DPA context

dpa_thread

- created DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NO_MEMORY - failure in internal memory allocation

Description

This function creates a DPA thread. DPA thread used to run a user kernel on DPA. User can control on which EU to run the DPA kernel. The thread is activated on DPA using two methods: 1- Thread Activation using DPA notification completion context. 2- Attaching completion context to the thread. To activate the thread in order to enable receiving any messages or completions, user must: 1- Create DPA thread and configure it using thread setter functions. 2- Attach to a completion context. 3- Set thread to runnable state using [doca_dpa_thread_run\(\)](#) API.

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_destroy (doca_dpa_thread
*dpa_thread)
```

Destroy DPA thread.

Parameters

dpa_thread

- Previously created DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA thread created by [doca_dpa_thread_create\(\)](#)

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_get_affinity (doca_dpa_thread
*dpa_thread, const doca_dpa_eu_affinity **affinity)
```

Get DPA thread affinity.

Parameters

dpa_thread

- DPA thread

affinity

- DPA EU affinity

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_get_func_arg (doca_dpa_thread
*dpa_thread, doca_dpa_func_t **func, uint64_t *arg)
```

Get DPA thread entry point and its argument.

Parameters

dpa_thread

- DPA thread

func

- DPA thread entry point

arg

- DPA thread entry point argument

Returns

- ▶ DOCA_SUCCESS - in case of success

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_get_local_storage
(doca_dpa_thread *dpa_thread,
 doca_dpa_dev_uintptr_t *dev_ptr)
```

Get DPA thread local storage.

Parameters

dpa_thread

- DPA thread

dev_ptr

- DPA device memory address

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_group_create (doca_dpa *dpa,
 unsigned int num_threads, doca_dpa_tg **tg)
```

Create DPA thread group.

Parameters

dpa

- DPA context

num_threads

- number of threads for the DPA thread group

tg

- created thread group

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_NO_MEMORY - failure in internal memory allocation

Description

This function creates an empty thread group (no populated threads) of a given size. User can set thread at specific rank using [doca_dpa_thread_group_set_thread\(\)](#) API. Threads must be set in all ranks before thread group is started

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_group_destroy (doca_dpa_tg *tg)

Destroy DPA thread group.

Parameters

tg

- Previously created DPA thread group

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

Description

This function destroys DPA thread group created by [doca_dpa_thread_group_create\(\)](#)

DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_group_get_num_threads
(doca_dpa_tg *tg, unsigned int *num_threads)

Get DPA thread group number of threads.

Parameters

tg

- DPA thread group

num_threads

- number of threads for the DPA thread group

Returns

- ▶ DOCA_SUCCESS - in case of success

- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input

DOCA_EXPERIMENTAL `doca_error_t`
`doca_dpa_thread_group_set_thread (doca_dpa_tg`
`*tg, doca_dpa_thread *thread, unsigned int rank)`

Set DPA thread at 'rank' in DPA thread group.

Parameters

tg

- DPA thread group

thread

- DPA thread

rank

- rank of the DPA thread in DPA thread group

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE

Description

Thread rank is an index of the thread (between 0 and size()-1) within the group. This function must be called before starting both DPA thread and DPA thread group

DOCA_EXPERIMENTAL `doca_error_t`
`doca_dpa_thread_group_start (doca_dpa_tg *tg)`

Start DPA thread group.

Parameters

tg

- DPA thread group

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

DOCA_EXPERIMENTAL doca_error_t doca_dpa_thread_group_stop (doca_dpa_tg *tg)

Stop DPA thread group.

Parameters

tg

- DPA thread group

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

DOCA_EXPERIMENTAL doca_error_t doca_dpa_thread_run (doca_dpa_thread *dpa_thread)

Run DPA thread.

Parameters

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

Description

This function sets the thread to runnable state such that when the completion context attached to the thread receives a message, the thread will run. This function must be called after DPA thread is: 1- Created and started. 2- Attached to a completion context.

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_set_affinity (doca_dpa_thread
*dpa_thread, const doca_dpa_eu_affinity *affinity)
```

Set DPA thread affinity.

Parameters

dpa_thread

- DPA thread

affinity

- DPA EU affinity

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA thread is already started

Description

This function sets the affinity type of DPA thread to "fixed". Fixed affinity means that the thread will run only on the EU ID provided in the given DPA EU affinity. When affinity is not specified, the default affinity mode is "relaxed", means the thread will run on any available EU ID when its rescheduled

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_set_func_arg (doca_dpa_thread
*dpa_thread, doca_dpa_func_t *func, uint64_t arg)
```

Set DPA thread entry point and its argument.

Parameters

dpa_thread

- DPA thread

func

- Host function pointer representing DPA kernel which thread run when it is triggered

arg

- DPA thread entry point argument

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA thread is already started

Description

The DPA thread function `func` is a host function pointer corresponding to the DPA device function. The device function must be annotated with `__dpa_global__` keyword such as `__dpa_global__ void hello(int arg1)`. In the Host application, the user is expected to declare the function as `extern doca_dpa_func_t hello;`. After the application is linked and loaded using the compiler, the function pointer `hello` can be used in as the `func` argument. This function must be called before DPA thread is started

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_thread_set_local_storage
(doca_dpa_thread *dpa_thread,
 doca_dpa_dev_uintptr_t dev_ptr)
```

Set DPA thread local storage.

Parameters

dpa_thread

- DPA thread

dev_ptr

- DPA device memory address

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_BAD_STATE - DPA thread is already started

Description

User can ask to store an opaque (DPA device memory pointer) for a DPA thread in host side using Thread local storage utility. In device kernel, user can obtain this opaque using `doca_dpa_dev_thread_get_local_storage()` API. This function must be called before DPA thread is started

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_thread_start (doca_dpa_thread  
*dpa_thread)
```

Start DPA thread.

Parameters

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE

```
DOCA_EXPERIMENTAL doca_error_t  
doca_dpa_thread_stop (doca_dpa_thread  
*dpa_thread)
```

Stop DPA thread.

Parameters

dpa_thread

- DPA thread

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid input
- ▶ DOCA_ERROR_DRIVER - error in a DOCA driver call
- ▶ DOCA_ERROR_BAD_STATE - attached DPA context is destroyed

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_trace_file_get_path (doca_dpa *dpa, char
*file_path, uint32_t *file_path_len)
```

Get trace file path to write device traces into.

Parameters

dpa

- DPA context

file_path

- pathname to the trace file to write device traces into

file_path_len

- file_path length. Output is actual number of bytes written

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid null input
- ▶ DOCA_ERROR_NO_MEMORY - memory allocation error

Description

The file_path buffer is allocated by the caller along with setting file_path_len indicating the length that was allocated. Upon return the file_path_len field is set to the actual length of the file_path

```
DOCA_EXPERIMENTAL doca_error_t
doca_dpa_trace_file_set_path (doca_dpa *dpa, const
char *file_path)
```

Set trace file path to write device traces into.

Parameters

dpa

- DPA context

file_path

- pathname to the trace file to write device traces into

Returns

- ▶ DOCA_SUCCESS - in case of success
- ▶ DOCA_ERROR_INVALID_VALUE - received invalid null input

- ▶ DOCA_ERROR_BAD_STATE - DPA context is already started
- ▶ DOCA_ERROR_NO_MEMORY - memory allocation error
- ▶ DOCA_ERROR_OPERATING_SYSTEM - error occurred in opening the file

Description

Must be set before calling [doca_dpa_start\(\)](#)

**Note:**

if not set then stdout will be used by default

#define

DOCA_DPA_COMPLETION_LOG_MAX_USER_DATA (24)

DPA completion Log max user data type definition.

Valid values are greater equal to 0 and less than (1 << DOCA_DPA_COMPLETION_LOG_MAX_USER_DATA)

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 NON-INFRINGEMENT, 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

© 2024 NVIDIA Corporation & affiliates. All rights reserved.