DOCA Libraries API

Reference Manual
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>doca_apsh_start</td>
<td>19</td>
</tr>
<tr>
<td>doca_apsh_sys_dev_set</td>
<td>20</td>
</tr>
<tr>
<td>doca_apsh_sys_kpgd_file_set</td>
<td>20</td>
</tr>
<tr>
<td>doca_apsh_sys_mem_region_set</td>
<td>21</td>
</tr>
<tr>
<td>doca_apsh_sys_os_symbol_map_set</td>
<td>22</td>
</tr>
<tr>
<td>doca_apsh_sys_os_type_set</td>
<td>22</td>
</tr>
<tr>
<td>doca_apsh_system_create</td>
<td>23</td>
</tr>
<tr>
<td>doca_apsh_system_destroy</td>
<td>23</td>
</tr>
<tr>
<td>doca_apsh_system_start</td>
<td>24</td>
</tr>
<tr>
<td>doca_apsh_threads_free</td>
<td>24</td>
</tr>
<tr>
<td>doca_apsh_threads_get</td>
<td>24</td>
</tr>
<tr>
<td>doca_apsh_vads_free</td>
<td>25</td>
</tr>
<tr>
<td>doca_apsh_vads_get</td>
<td>25</td>
</tr>
<tr>
<td>doca_apsh_attst_info_get</td>
<td>26</td>
</tr>
<tr>
<td>doca_apsh_envar_info_get</td>
<td>26</td>
</tr>
<tr>
<td>doca_apsh_handle_info_get</td>
<td>27</td>
</tr>
<tr>
<td>doca_apsh_ldrmodule_info_get</td>
<td>27</td>
</tr>
<tr>
<td>doca_apsh_lib_info_get</td>
<td>27</td>
</tr>
<tr>
<td>doca_apsh_module_info_get</td>
<td>27</td>
</tr>
<tr>
<td>doca_apsh_privilege_info_get</td>
<td>27</td>
</tr>
<tr>
<td>doca_apsh_process_info_get</td>
<td>28</td>
</tr>
<tr>
<td>doca_apsh_sys_config</td>
<td>28</td>
</tr>
<tr>
<td>doca_apsh_thread_info_get</td>
<td>28</td>
</tr>
<tr>
<td>doca_apsh_vad_info_get</td>
<td>28</td>
</tr>
</tbody>
</table>

2.2. App Shield Attributes

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>doca_apsh_attestation_attr</td>
<td>29</td>
</tr>
<tr>
<td>doca_apsh_envar_attr</td>
<td>29</td>
</tr>
<tr>
<td>doca_apsh_handle_attr</td>
<td>30</td>
</tr>
<tr>
<td>doca_apsh_ldrmodule_attr</td>
<td>30</td>
</tr>
<tr>
<td>doca_apsh_lib_attr</td>
<td>31</td>
</tr>
<tr>
<td>doca_apsh_module_attr</td>
<td>31</td>
</tr>
<tr>
<td>doca_apsh_privilege_attr</td>
<td>31</td>
</tr>
<tr>
<td>doca_apsh_process_attr</td>
<td>32</td>
</tr>
<tr>
<td>doca_apsh_system_config_attr</td>
<td>32</td>
</tr>
<tr>
<td>doca_apsh_system_os</td>
<td>33</td>
</tr>
<tr>
<td>doca_apsh_thread_attr</td>
<td>33</td>
</tr>
<tr>
<td>doca_apsh_vad_attr</td>
<td>34</td>
</tr>
<tr>
<td>DOCA_APSH_ATTESTATION_COMM_TYPE</td>
<td>34</td>
</tr>
</tbody>
</table>
DOCA_APSH_ATTESTATION_END_ADDRESS_TYPE.................................................................34
DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT_TYPE..............................................35
DOCA_APSH_ATTESTATION_MATCHING_HASHES_TYPE..................................................35
DOCA_APSH_ATTESTATION_PAGES_NUMBER_TYPE..........................................................35
DOCA_APSH_ATTESTATION_PAGES_PRESENT_TYPE.........................................................35
DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA_TYPE.................................35
DOCA_APSH_ATTESTATION_PID_TYPE...........................................................................35
DOCA_APSH_ATTESTATION_PROTECTION_TYPE.............................................................35
DOCA_APSH_ATTESTATION_START_ADDRESS_TYPE.......................................................35
DOCA_APSH_DMA_DEV_TYPE.........................................................................................35
DOCA_APSH_ENVARS_COMM_TYPE................................................................................36
DOCA_APSH_ENVARS_PID_TYPE....................................................................................36
DOCA_APSH_ENVARS_VALUE_TYPE................................................................................36
DOCA_APSH_ENVARS_VARIABLE_TYPE..........................................................................36
DOCA_APSH_ENVARS_WINDOWS_BLOCK_TYPE.............................................................36
DOCA_APSH_HANDLE_ACCESS_TYPE...............................................................................36
DOCA_APSH_HANDLE_COMM_TYPE...............................................................................36
DOCA_APSH_HANDLE_NAME_TYPE................................................................................36
DOCA_APSH_HANDLE_PID_TYPE....................................................................................36
DOCA_APSH_HANDLE_TABLE_ENTRY_TYPE.................................................................36
DOCA_APSH_HANDLE_TYPE_TYPE................................................................................36
DOCA_APSH_HANDLE_VALUE_TYPE................................................................................36
DOCA_APSH_HASHTEST_LIMIT_TYPE..............................................................................37
DOCA_APSH_KPGD_FILE_TYPE.......................................................................................37
DOCA_APSH_LDRMODULE_BASE_ADDRESS_TYPE...........................................................37
DOCA_APSH_LDRMODULE_COMM_TYPE.........................................................................37
DOCA_APSH_LDRMODULE_LIBRARY_PATH_TYPE............................................................37
DOCA_APSH_LDRMODULE_PID_TYPE...............................................................................37
DOCA_APSH_LDRMODULE_WINDOWS_BASE_DLL_NAME_TYPE..........................................37
DOCA_APSH_LDRMODULE_WINDOWS_ININIT_TYPE........................................................37
DOCA_APSH_LDRMODULE_WINDOWS_INLOAD_TYPE.......................................................38
DOCA_APSH_LDRMODULE_WINDOWS_INMEM_TYPE.........................................................38
DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE_TYPE...........................................38
DOCA_APSH_LIB_COMM_TYPE....................................................................................38
DOCA_APSH_LIB_LIBRARY_PATH_TYPE.........................................................................38
DOCA_APSH_LIB_LINUX_LOAD_ADDRESS_TYPE............................................................38
DOCA_APSH_LIB_PID_TYPE.........................................................................................38
DOCA_APSH_LIB_WINDOWS_FULL_DLL_NAME_TYPE.........................................................38
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCA_APSH_LIB_WINDOWS_SIZE_OFIMAGE_TYPE</td>
<td>38</td>
</tr>
<tr>
<td>DOCA_APSH_LIBS_LIMIT_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_MEM_REGION_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_MODULES_LIMIT_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_MODULES_NAME_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_MODULES_OFFSET_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_MODULES_SIZE_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_OS_SYMBOL_MAP_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_OS_TYPE_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_COMM_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_IS_ON_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_NAME_TYPE</td>
<td>39</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_PID_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_COMM_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_CPU_TIME_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_LIMIT_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_LINUX_GID_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_LINUX_UID_TYPE</td>
<td>40</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_PID_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_PPID_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_STATE_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_WINDOWS_OFFSET_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_PROCESS_WINDOWS_THREADS_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_REGEX_DEV_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_STRING_LIMIT_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_LINUX_PROC_NAME_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_LINUX_THREAD_NAME_TYPE</td>
<td>41</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_PID_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_STATE_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_TID_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_WINDOWS_OFFSET_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_THREAD_WINDOWS_WAIT_REASON_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_THREADS_LIMIT_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_VADS_LIMIT_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>DOCA_APSH_VHCA_ID_TYPE</td>
<td>42</td>
</tr>
<tr>
<td>Documentation Section</td>
<td>Reference Page</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>DoCA APSH VMA File Path Type</td>
<td>42</td>
</tr>
<tr>
<td>DoCA APSH VMA Offset Type</td>
<td>42</td>
</tr>
<tr>
<td>DoCA APSH VMA PID Type</td>
<td>42</td>
</tr>
<tr>
<td>DoCA APSH VMA Process Name Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH VMA Protection Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH VMA VM End Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH VMA VM Start Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH Windows Commit Charge Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH Windows Private Memory Type</td>
<td>43</td>
</tr>
<tr>
<td>DoCA APSH Windows ENVARS Limit Type</td>
<td>43</td>
</tr>
<tr>
<td>2.4. Core</td>
<td>43</td>
</tr>
<tr>
<td>DoCA Buffer</td>
<td>43</td>
</tr>
<tr>
<td>DoCA Buffer Inventory</td>
<td>43</td>
</tr>
<tr>
<td>DoCA Context</td>
<td>43</td>
</tr>
<tr>
<td>DoCA Device</td>
<td>43</td>
</tr>
<tr>
<td>DoCA DPDK</td>
<td>44</td>
</tr>
<tr>
<td>DoCA Error</td>
<td>44</td>
</tr>
<tr>
<td>DoCA Hotplug</td>
<td>44</td>
</tr>
<tr>
<td>DoCA Memory Map</td>
<td>44</td>
</tr>
<tr>
<td>DoCA RDMA</td>
<td>44</td>
</tr>
<tr>
<td>DoCA Types</td>
<td>44</td>
</tr>
<tr>
<td>2.4.1. DoCA Buffer</td>
<td>44</td>
</tr>
<tr>
<td><code>doca_buf_get_data</code></td>
<td>44</td>
</tr>
<tr>
<td><code>doca_buf_get_data_len</code></td>
<td>45</td>
</tr>
<tr>
<td><code>doca_buf_get_head</code></td>
<td>45</td>
</tr>
<tr>
<td><code>doca_buf_get_len</code></td>
<td>45</td>
</tr>
<tr>
<td><code>doca_buf_get_lkey</code></td>
<td>46</td>
</tr>
<tr>
<td><code>doca_buf_get_refcount</code></td>
<td>46</td>
</tr>
<tr>
<td><code>doca_buf_list_chain</code></td>
<td>47</td>
</tr>
<tr>
<td><code>doca_buf_list_is_first</code></td>
<td>47</td>
</tr>
<tr>
<td><code>doca_buf_list_is_last</code></td>
<td>48</td>
</tr>
<tr>
<td><code>doca_buf_last</code></td>
<td>48</td>
</tr>
<tr>
<td><code>doca_buf_next</code></td>
<td>48</td>
</tr>
<tr>
<td><code>doca_buf_num_elements</code></td>
<td>49</td>
</tr>
<tr>
<td><code>doca_buf_unchain</code></td>
<td>49</td>
</tr>
<tr>
<td><code>doca_buf_refcount_add</code></td>
<td>50</td>
</tr>
<tr>
<td><code>doca_buf_refcount_rm</code></td>
<td>50</td>
</tr>
<tr>
<td><code>doca_buf_set_data</code></td>
<td>51</td>
</tr>
</tbody>
</table>
2.4.2. DOCA Buffer Inventory..................................................................................................... 51
    doca_buf_inventory_buf_by_addr........................................................................................... 52
    doca_buf_inventory_buf_by_args........................................................................................... 52
    doca_buf_inventory_buf_by_data........................................................................................... 53
    doca_buf_inventory_buf_dup..................................................................................................54
    doca_buf_inventory_create.....................................................................................................54
    doca_buf_inventory_destroy................................................................................................... 55
    doca_buf_inventory_get_list_supported................................................................................ 55
    doca_buf_inventory_get_num_elements............................................................................... 56
    doca_buf_inventory_get_num_free_elements....................................................................... 56
    doca_buf_inventory_get_user_data....................................................................................... 57
    doca_buf_inventory_start........................................................................................................57
    doca_buf_inventory_stop........................................................................................................ 58

2.4.3. DOCA Context................................................................................................................... 58
    doca_event...............................................................................................................................59
    doca_job...................................................................................................................................59
    doca_ctx_dev_add................................................................................................................... 59
    doca_ctx_dev_rm.....................................................................................................................59
    doca_ctx_get_event_driven_supported..................................................................................60
    doca_ctx_get_max_num_ctx.................................................................................................. 60
    doca_ctx_start......................................................................................................................... 61
    doca_ctx_stop..........................................................................................................................61
    doca_ctx_workq_add...............................................................................................................62
    doca_ctx_workq_rm................................................................................................................62
    doca_workq_create.................................................................................................................63
    doca_workq_destroy............................................................................................................... 63
    doca_workq_event_handle_arm.............................................................................................64
    doca_workq_event_handle_clear...........................................................................................64
    doca_workq_get_depth........................................................................................................... 65
    doca_workq_get_event_driven_enable.................................................................................. 65
    doca_workq_get_event_handle.............................................................................................. 66
    doca_workq_progress_retrieve..............................................................................................66
    doca_workq_set_depth........................................................................................................... 67
    doca_workq_set_event_driven_enable................................................................................... 67
    doca_workq_submit................................................................................................................ 68
    DOCA_ACTION_SDK_RANGE................................................................................................. 68

2.4.4. DOCA Device..................................................................................................................... 69
    doca_dev_rep_filter.................................................................................................................69
doca_dev_as_devinfo...............................................................................................................69
doca_dev_close....................................................................................................................... 69
doca_dev_open........................................................................................................................ 70
doca_dev_rep_as_devinfo....................................................................................................... 70
doca_dev_rep_close................................................................................................................71
doca_dev_rep_open.................................................................................................................71
doca_devinfo_get_ibdev_name...............................................................................................71
doca_devinfo_get_iface_name................................................................................................72
doca_devinfo_get_ipv4_addr...................................................................................................72
doca_devinfo_get_ipv6_addr...................................................................................................73
doca_devinfo_get_is_hotplug_manager_supported..............................................................74
doca_devinfo_get_is_mmap_export_supported.................................................................... 74
doca_devinfo_get_is_mmap_from_export_supported.......................................................... 75
doca_devinfo_get_pci_addr.................................................................................................... 75
doca_devinfo_list_create........................................................................................................ 76
doca_devinfo_list_destroy.......................................................................................................76
doca_devinfo_rep_get_is_list_all_supported........................................................................ 77
doca_devinfo_rep_get_is_list_net_supported....................................................................... 78
doca_devinfo_rep_get_is_list_nvme_supported....................................................................78
doca_devinfo_rep_get_is_list_virtio_blk_supported............................................................. 79
doca_devinfo_rep_get_is_list_virtio_fs_supported............................................................... 80
doca_devinfo_rep_get_is_list_virtio_net_supported............................................................. 80
doca_devinfo_rep_get_pci_addr.............................................................................................81
doca_devinfo_rep_get_pci_func_type.................................................................................... 81
doca_devinfo_rep_get_vuid.....................................................................................................82
doca_devinfo_rep_list_create.................................................................................................82
doca_devinfo_rep_list_destroy............................................................................................... 83
2.4.5. DOCA DPDK...................................................................................................................... 84
doca_dpdk_port_as_dev......................................................................................................... 84
doca_dpdk_port_probe........................................................................................................... 84
2.4.6. DOCA Error....................................................................................................................... 85
doca_get_error_name.............................................................................................................85
doca_get_error_string............................................................................................................ 85
DOCA_ERROR_PROPAGATE...................................................................................................86
2.4.7. DOCA Hotplug................................................................................................................... 86
doca_dev_rep_hotplug............................................................................................................ 86
doca_dev_rep_hotunplug.........................................................................................................86
2.4.8. DOCA Memory Map.......................................................................................................... 87
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>doca_mmap_memrange_free_cb_t</td>
<td>87</td>
</tr>
<tr>
<td>doca_mmap_create</td>
<td>87</td>
</tr>
<tr>
<td>doca_mmap_create_from_export</td>
<td>88</td>
</tr>
<tr>
<td>doca_mmap_destroy</td>
<td>89</td>
</tr>
<tr>
<td>doca_mmap_dev_add</td>
<td>89</td>
</tr>
<tr>
<td>doca_mmap_dev_rm</td>
<td>90</td>
</tr>
<tr>
<td>doca_mmap_export</td>
<td>91</td>
</tr>
<tr>
<td>doca_mmap_get_exported</td>
<td>92</td>
</tr>
<tr>
<td>doca_mmap_get_from_export</td>
<td>92</td>
</tr>
<tr>
<td>doca_mmap_get_max_num_chunks</td>
<td>93</td>
</tr>
<tr>
<td>doca_mmap_get_max_num_devices</td>
<td>93</td>
</tr>
<tr>
<td>doca_mmap_get_num_bufs</td>
<td>93</td>
</tr>
<tr>
<td>doca_mmap_get_user_data</td>
<td>94</td>
</tr>
<tr>
<td>doca_mmap_populate</td>
<td>94</td>
</tr>
<tr>
<td>doca_mmap_set_max_num_chunks</td>
<td>95</td>
</tr>
<tr>
<td>doca_mmap_set_max_num_devices</td>
<td>96</td>
</tr>
<tr>
<td>doca_mmap_start</td>
<td>96</td>
</tr>
<tr>
<td>doca_mmap_stop</td>
<td>97</td>
</tr>
<tr>
<td>2.4.9. DOCA RDMA</td>
<td>97</td>
</tr>
<tr>
<td>doca_dev_get_pd</td>
<td>97</td>
</tr>
<tr>
<td>2.4.10. DOCA Types</td>
<td>98</td>
</tr>
<tr>
<td>doca_pci_bdf</td>
<td>98</td>
</tr>
<tr>
<td>doca_access_flags</td>
<td>98</td>
</tr>
<tr>
<td>doca_pci_func_type</td>
<td>98</td>
</tr>
<tr>
<td>2.5. Comm Channel</td>
<td>98</td>
</tr>
<tr>
<td>doca_comm_channel_msg_flags</td>
<td>98</td>
</tr>
<tr>
<td>doca_event_channel_t</td>
<td>99</td>
</tr>
<tr>
<td>doca_comm_channel_ep_connect</td>
<td>99</td>
</tr>
<tr>
<td>doca_comm_channel_ep_create</td>
<td>100</td>
</tr>
<tr>
<td>doca_comm_channel_ep_destroy</td>
<td>100</td>
</tr>
<tr>
<td>doca_comm_channel_ep_disconnect</td>
<td>101</td>
</tr>
<tr>
<td>doca_comm_channel_ep_event_handle_arm_recv</td>
<td>101</td>
</tr>
<tr>
<td>doca_comm_channel_ep_get_device</td>
<td>102</td>
</tr>
<tr>
<td>doca_comm_channel_ep_get_device_rep</td>
<td>102</td>
</tr>
<tr>
<td>doca_comm_channel_ep_get_event_channel</td>
<td>103</td>
</tr>
<tr>
<td>doca_comm_channel_ep_get_max_msg_size</td>
<td>103</td>
</tr>
<tr>
<td>doca_comm_channel_ep_get_recv_queue_size</td>
<td>104</td>
</tr>
</tbody>
</table>
2.6. Compatibility Management

__DOCA_EXPERIMENTAL__

DOCA_STRUCT_START

2.7. DOCA COMPRESS engine

doca_compress_job...

doca_compress_job_types...

doca_compress_as_ctx...

doca_compress_create...

doca_compress_destroy...

doca_compress_get_max_buffer_size...

doca_compress_get_max_list_buf_num_elem...

doca_compress_job_get_supported...

2.8. Environment Configurations

DOCA_COMPAT_HELPERS

2.9. ct...

doca_ct_cfg...
DOCA Libraries API

2.13. flow net define

2.14. Flow
`doca_flow_grpc_port_destroy`.................................................................................................. 174
`doca_flow_grpc_port_pair`........................................................................................................174
`doca_flow_grpc_port_pipes_dump`.......................................................................................... 174
`doca_flow_grpc_port_pipes_flush`........................................................................................... 175
`doca_flow_grpc_port_start`.......................................................................................................175
`doca_flow_grpc_port_stop`....................................................................................................... 175
`doca_flow_grpc_port_switch_get`............................................................................................ 175
`doca_flow_grpc_query`..............................................................................................................176
`doca_flow_grpc_shared_resource_cfg`....................................................................................176
`doca_flow_grpc_shared_resources_bind`................................................................................177
`doca_flow_grpc_shared_resources_query`..............................................................................177

2.15. flow net define.....................................................................................................................178
`doca_flow_ip_addr`.................................................................................................................... 178
`doca_flow_tun`........................................................................................................................... 178
`doca_flow_ip_type`.....................................................................................................................178
`doca_flow_tun_type`.................................................................................................................. 178
`doca_be16_t`.............................................................................................................................. 179
`doca_be32_t`.............................................................................................................................. 179
`doca_be64_t`.............................................................................................................................. 179
`DOCA_ETHER_ADDR_LEN`.......................................................................................................179
`DOCA_ETHER_TYPE_IPV4`........................................................................................................ 179
`DOCA_ETHER_TYPE_IPV6`........................................................................................................ 179
`DOCA_ETHER_TYPE_TEB`.........................................................................................................179

2.16. IPsec.....................................................................................................................................181
`doca_encryption_key`.................................................................................................................181
2.17. Logging Management

- doca_log_registrator
- doca_log_level
- log_flush_callback
- doca_log
- doca_log_backend_level_set
- doca_log_create_buffer_backend
- doca_log_create_fd_backend
- doca_log_create_file_backend
- doca_log_create_syslog_backend
- doca_log_developer
- doca_log_get_bucket_time
- doca_log_get_quantity
- doca_log_global_level_get
- doca_log_global_level_set
- doca_log_rate_bucket_register
- doca_log_rate_limit
- doca_log_set_bucket_time
doca_log_set_quantity.............................................................................................................. 193
doca_log_source_destroy.........................................................................................................194
doca_log_source_register........................................................................................................ 194
doca_log_stream_redirect....................................................................................................... 195
DOCA_DLOG.............................................................................................................................. 195
DOCA_DLOG_CRIT.................................................................................................................... 195
DOCA_DLOG_DBG.................................................................................................................... 195
DOCA_DLOG_ERR.....................................................................................................................196
DOCA_DLOG_INFO....................................................................................................................196
DOCA_DLOG_WARN....................................................................................................................196
DOCA_LOG.................................................................................................................................196
DOCA_LOG_CRIT.......................................................................................................................197
DOCA_LOG_DBG....................................................................................................................... 197
DOCA_LOG_ERR....................................................................................................................... 197
DOCA_LOG_INFO....................................................................................................................... 197
DOCA_LOG_RATE_LIMIT........................................................................................................... 198
DOCA_LOG_RATE_LIMIT_CRIT................................................................................................ 198
DOCA_LOG_RATE_LIMIT_DBG.................................................................................................198
DOCA_LOG_RATE_LIMIT_ERR................................................................................................. 198
DOCA_LOG_RATE_LIMIT_INFO................................................................................................198
DOCA_LOG_RATE_LIMIT_WARN..............................................................................................199
DOCA_LOG_WARN....................................................................................................................199
2.18. RegEx engine.......................................................................................................................199
doca_regex_job_search............................................................................................................ 199
doca_regex_match....................................................................................................................199
doca_regex_search_result....................................................................................................... 199
doca_regex_job_types............................................................................................................. 199
doca_regex_search_job_flags..................................................................................................199
doca_regex_status_flag............................................................................................................200
doca_regex_as_ctx....................................................................................................................200
doca_regex_create....................................................................................................................200
doca_regex_destroy..................................................................................................................201
doca_regex_get_failed_job_fallback_enabled.........................................................................201
doca_regex_get_hardware_compiled_rules........................................................................... 202
doca_regex_get_hardware_supported.................................................................................... 202
doca_regex_get_hardware_uncompiled_rules....................................................................... 203
doca_regex_get_huge_job_emulation_overlap_size...............................................................203
doca_regex_get_maximum_job_size....................................................................................... 204
doca_regex_get_max_mimum_non_huge_job_size.....................................................................204

2.19. RegEx engine memory pool............................................................................................216
doca_regex_mempool_create.................................................................................................. 216
doca_regex_mempool_destroy................................................................................................. 217
doca_regex_mempool_get_nth_element................................................................................ 217
doca_regex_mempool_get_obj................................................................................................ 218
doca_regex_mempool_index_of............................................................................................... 218
doca_regex_mempool_put_obj................................................................................................. 219

2.20. DOCA RMAX engine........................................................................................................219
doca_rmax_cpu_affinity_mask................................................................................................. 219
doca_rmax_in_stream_completion........................................................................................... 219
doca_rmax_stream_error......................................................................................................... 219
doca_rmax_action_type............................................................................................................ 219
doca_rmax_in_stream_scatter_type........................................................................................ 220
doca_rmax_in_stream_ts_fmt_type......................................................................................... 220
doca_rmax_in_stream_type..................................................................................................... 220
doca_rmax_cpu_mask_t........................................................................................................... 220
doca_rmax_flow_attach............................................................................................................ 221
doca_rmax_flow_create............................................................................................................ 221
doca_rmax_flow_destroy.......................................................................................................... 222
doca_rmax_flowDetach................................................................................................................222
doca_rmax_flow_set_dst_ip .......................................................... 222
doca_rmax_flow_set_dst_port ................................................. 223
doca_rmax_flow_set_src_ip .................................................... 223
doca_rmax_flow_set_src_port ................................................. 224
doca_rmax_flow_set_tag ......................................................... 224
doca_rmax_get_cpu_affinity_mask ........................................ 224
doca_rmax_get_ptp_clock_supported ..................................... 225
doca_rmax_in_stream_as_ctx .................................................. 225
doca_rmax_in_stream_create .................................................. 226
doca_rmax_in_stream_destroy ............................................... 226
doca_rmax_in_stream_get_elements_count ........................... 227
doca_rmax_in_stream_get_max_packets ................................. 227
doca_rmax_in_stream_get_memblk_size ................................. 228
doca_rmax_in_stream_get_memblk_stride_size ..................... 228
doca_rmax_in_stream_get_memblks_count ............................. 229
doca_rmax_in_stream_get_min_packets ................................. 229
doca_rmax_in_stream_get_scatter_type ................................. 230
doca_rmax_in_stream_get_timeout_us ................................. 230
doca_rmax_in_stream_get_timestamp_format ....................... 231
doca_rmax_in_stream_get_type ............................................. 231
doca_rmax_in_stream_memblk_desc_get_max_size .................. 232
doca_rmax_in_stream_memblk_desc_get_min_size .................. 232
doca_rmax_in_stream_memblk_desc_set_max_size .................. 233
doca_rmax_in_stream_memblk_desc_set_min_size .................. 234
doca_rmax_in_stream_set_elements_count .......................... 234
doca_rmax_in_stream_set_max_packets ............................... 235
doca_rmax_in_stream_set_memblk ........................................ 235
doca_rmax_in_stream_set_memblks_count ............................ 236
doca_rmax_in_stream_set_min_packets ............................... 236
doca_rmax_in_stream_set_scatter_type ............................... 237
doca_rmax_in_stream_set_timeout_us ............................... 237
doca_rmax_in_stream_set_timestamp_format ....................... 238
doca_rmax_in_stream_set_type ........................................... 239
doca_rmax_init ................................................................. 239
doca_rmax_interrupt ......................................................... 240
doca_rmax_release ............................................................ 240
doca_rmax_set_clock ........................................................ 240
doca_rmax_set_cpu_affinity_mask ....................................... 241
2.21. engine

- doca_sha_job
- doca_sha_partial_job
- doca_sha_job_flags
- doca_sha_job_type
- doca_sha_as_ctx
- doca_sha_create
- doca_sha_destroy
- doca_sha_get_hardware_supported
- doca_sha_get_max_list_buf_num_elem
- doca_sha_get_max_src_buffer_size
- doca_sha_get_min_dst_buffer_size
- doca_sha_job_get_supported
- doca_sha_partial_session_copy
- doca_sha_partial_session_create
- doca_sha_partial_session_destroy
- DOCA_SHA1_BYTE_COUNT
- DOCA_SHA256_BYTE_COUNT
- DOCA_SHA512_BYTE_COUNT

2.22. Telemetry Service Library

- doca_telemetry_ipc_status_t
- doca_guid_t
- doca_telemetry_timestamp_t
- doca_telemetry_type_index_t
- doca_telemetry_check_ipc_status
- doca_telemetry_field_create
- doca_telemetry_field_destroy
- doca_telemetry_field_set_array_length
- doca_telemetry_field_set_description
- doca_telemetry_field_set_name
- doca_telemetry_field_set_type_name
- doca_telemetry_get_timestamp
- doca_telemetry_netflow_destroy
- doca_telemetry_netflow_field_create
- doca_telemetry_netflow_field_destroy
- doca_telemetry_netflow_field_set_length
doca_telemetry_netflow_field_set_type.................................................................................. 255
doca_telemetry_netflow_flush................................................................................................. 256
doca_telemetry_netflow_get_buffer_data_root....................................................................... 256
doca_telemetry_netflow_get_buffer_size................................................................................256
doca_telemetry_netflow_get_file_write_max_age..................................................................257
doca_telemetry_netflow_get_file_write_max_size..................................................................257
doca_telemetry_netflow_get_ipc_sockets_dir........................................................................ 257
doca_telemetry_netflow_init.................................................................................................... 258
doca_telemetry_netflow_send................................................................................................. 259
doca_telemetry_netflow_set_buffer_data_root...................................................................... 260
doca_telemetry_netflow_set_buffer_size................................................................................260
doca_telemetry_netflow_set_collector_addr.......................................................................... 261
doca_telemetry_netflow_set_collector_port...........................................................................261
doca_telemetry_netflow_set_file_write_enabled....................................................................262
doca_telemetry_netflow_set_file_write_max_age..................................................................262
doca_telemetry_netflow_set_file_write_max_size..................................................................262
doca_telemetry_netflow_set_ipc_enabled.............................................................................. 263
doca_telemetry_netflow_set_ipc_sockets_dir........................................................................ 263
doca_telemetry_netflow_set_max_packet_size......................................................................264
doca_telemetry_netflow_source_set_id.................................................................................. 264
doca_telemetry_netflow_source_set_tag................................................................................265
doca_telemetry_netflow_start..................................................................................................265
doca_telemetry_netflow_template_add_field......................................................................... 265
doca_telemetry_netflow_template_create.............................................................................. 266
doca_telemetry_netflow_template_destroy............................................................................ 267
doca_telemetry_schema_add_type..........................................................................................267
doca_telemetry_schema_destroy............................................................................................ 268
doca_telemetry_schema_get_buffer_data_root..................................................................... 268
doca_telemetry_schema_get_buffer_size...............................................................................268
doca_telemetry_schema_get_file_write_max_age..................................................................269
doca_telemetry_schema_get_file_write_max_size..................................................................269
doca_telemetry_schema_get_ipc_reconnect_time................................................................ 270
doca_telemetry_schema_get_ipc_reconnect_tries................................................................ 271
doca_telemetry_schema_get_ipc_socket_timeout................................................................... 271
doca_telemetry_schema_get_ipc_sockets_dir....................................................................... 272
doca_telemetry_schema_init................................................................................................... 272
doca_telemetry_schema_set_buffer_data_root......................................................................273
doca_telemetry_schema_set_buffer_size............................................................................... 273
doctelemetry_schema_set_file_write_enabled................................................................. 274
doctelemetry_schema_set_file_write_max_age........................................................... 274
doctelemetry_schema_set_file_write_max_size........................................................... 275
doctelemetry_schema_set_ipc_enabled.................................................................275
doctelemetry_schema_set_ipc_reconnect_time......................................................276
doctelemetry_schema_set_ipc_reconnect_tries.......................................................276
doctelemetry_schema_set_ipc_socket_timeout......................................................277
doctelemetry_schema_set_ipc_sockets_dir.............................................................. 277
 doctelemetry_schema_set_opaque_events_enabled.................................................. 278
doctelemetry_schema_start...................................................................................... 278
 doctelemetry_source_create................................................................................... 279
doctelemetry_source_destroy.................................................................................. 279
 doctelemetry_source_flush..................................................................................... 280
doctelemetry_source_get_opaque_report_max_data_size.......................................280
doctelemetry_source_opaque_report.......................................................................281
doctelemetry_source_report................................................................................... 282
doctelemetry_source_set_id..................................................................................... 282
doctelemetry_source_set_tag................................................................................... 283
doctelemetry_source_start...................................................................................... 283
doctelemetry_type_add_field................................................................................... 284
doctelemetry_type_create....................................................................................... 284
 doctelemetry_type_destroy..................................................................................... 285
 DOCA_GUID_SIZE...................................................................................................... 285
DOCA_NETFLOW_APP_ID............................................................................................ 285
DOCA_NETFLOW_DEFAULT_PORT................................................................................285
DOCA_TELEMETRY_FIELD_TYPE_BOOL...................................................................... 286
DOCA_TELEMETRY_FIELD_TYPE_CHAR....................................................................... 286
DOCA_TELEMETRY_FIELD_TYPE_DOUBLE.....................................................................286
DOCA_TELEMETRY_FIELD_TYPE_FLOAT...................................................................... 286
DOCA_TELEMETRY_FIELD_TYPE_IN............................................................................ 286
DOCA_TELEMETRY_FIELD_TYPE_INT16.......................................................................286
DOCA_TELEMETRY_FIELD_TYPE_INT32....................................................................... 286
DOCA_TELEMETRY_FIELD_TYPE_INT64........................................................................ 286
DOCA_TELEMETRY_FIELD_TYPE_INT8......................................................................... 287
DOCA_TELEMETRY_FIELD_TYPE_LONG.......................................................................... 287
DOCA_TELEMETRY_FIELD_TYPE_LONGLONG.................................................................. 287
DOCA_TELEMETRY_FIELD_TYPE_SHORT....................................................................... 287
DOCA_TELEMETRY_FIELD_TYPE_TIMESTAMP................................................................ 287
DOCA_TELEMETRY_FIELD_TYPE_UCHAR .............................................................................. 287
DOCA_TELEMETRY_FIELD_TYPE_UINT .................................................................................. 287
DOCA_TELEMETRY_FIELD_TYPE_UINT16.............................................................................. 287
DOCA_TELEMETRY_FIELD_TYPE_UINT32.............................................................................. 288
DOCA_TELEMETRY_FIELD_TYPE_UINT64.............................................................................. 288
DOCA_TELEMETRY_FIELD_TYPE_UINT8................................................................................ 288
DOCA_TELEMETRY_FIELD_TYPE_ULONG.............................................................................. 288
DOCA_TELEMETRY_FIELD_TYPE_ULONGLONG.................................................................... 288
DOCA_TELEMETRY_FIELD_TYPE_USHORT............................................................................ 288

2.23. Version Management........................................................................................................ 288
doca_version............................................................................................................................. 288
doca_version_runtime.............................................................................................................. 289
DOCA_CURRENT_VERSION_NUM........................................................................................... 289
DOCA_VER_MAJOR................................................................................................................... 289
DOCA_VER_MINOR................................................................................................................... 289
DOCA_VER_PATCH................................................................................................................... 289
DOCA_VER_STRING.................................................................................................................. 289
DOCA_VERSION_EQ_CURRENT............................................................................................... 290
DOCA_VERSION_LTE_CURRENT............................................................................................... 290
DOCA_VERSION_NUM.............................................................................................................. 290

2.3. Change Log............................................................................................................................ 290

Chapter 3. Data Structures........................................................................................................... 292
doca_compress_job...................................................................................................................... 294
base........................................................................................................................................... 295
dst_buff...................................................................................................................................... 295
output_chksum......................................................................................................................... 295
src_buff...................................................................................................................................... 295
doca_ct_cfg............................................................................................................................... 295
flags........................................................................................................................................... 295
ib_dev......................................................................................................................................... 295
ib_pd.......................................................................................................................................... 295
nb_queues................................................................................................................................. 295
doca_dma_job_memcpy............................................................................................................... 296
base........................................................................................................................................... 296
dst_buff...................................................................................................................................... 296
src_buff...................................................................................................................................... 296
doca_dma_memcpy_result.......................................................................................................... 296
result.......................................................................................................................................... 296
doca_dpi_config_t......................................................................................................................... 296
    max_packets_per_queue........................................................................................................ 296
    max_sig_match_len............................................................................................................ 296
    nb_queues......................................................................................................................... 297
    server_address................................................................................................................ 297

doca_dpi_grpc_generic_packet................................................................................................... 297
    seg_len.................................................................................................................................. 297
    segment.............................................................................................................................. 297

doca_dpi_grpc_result.................................................................................................................... 297
    info...................................................................................................................................... 297
    matched.............................................................................................................................. 297
    pkt........................................................................................................................................ 298
    status_flags....................................................................................................................... 298
    user_data.......................................................................................................................... 298

doca_dpi_parsing_info.................................................................................................................. 298
    dst_ip................................................................................................................................... 298
    dst_ip................................................................................................................................... 298
    ethertype............................................................................................................................ 298
    ipv4..................................................................................................................................... 298
    ipv6..................................................................................................................................... 299
    l4_dport............................................................................................................................... 299
    l4_protocol......................................................................................................................... 299
    l4_sport............................................................................................................................... 299
    src_ip.................................................................................................................................. 299
    src_ip.................................................................................................................................. 299

doca_dpi_result............................................................................................................................ 299
    info...................................................................................................................................... 299
    matched.............................................................................................................................. 299
    pkt........................................................................................................................................ 300
    status_flags....................................................................................................................... 300
    user_data.......................................................................................................................... 300

doca_dpi_sig_data........................................................................................................................ 300
    name..................................................................................................................................... 300
    sig_id................................................................................................................................. 300

doca_dpi_sig_info......................................................................................................................... 300
    action................................................................................................................................. 300
    sig_id................................................................................................................................. 300

doca_dpi_stat_info........................................................................................................................ 300
nb_http_parser_based..............................................................................................................301
nb_matches...............................................................................................................................301
nb_other_l4............................................................................................................................... 301
nb_other_l7............................................................................................................................... 301
nb_scanned_pkts...................................................................................................................... 301
nb_ssl_parser_based................................................................................................................301
nb_tcp_based............................................................................................................................ 301
nb_udp_based........................................................................................................................... 301
doca_encryption_key.................................................................................................................... 301
  implicit_iv.................................................................................................................................301
  raw_key.................................................................................................................................302
  salt...........................................................................................................................................302
  type..........................................................................................................................................302
doca_event.................................................................................................................................... 302
  result......................................................................................................................................302
  type..........................................................................................................................................302
  user_data................................................................................................................................302
doca_flow_action_desc...............................................................................................................302
  type............................................................................................................................................302
doca_flow_action_descs..............................................................................................................303
  dst_ip......................................................................................................................................303
  dst_mac...................................................................................................................................303
  dst_port..................................................................................................................................303
  eth_type..................................................................................................................................303
  meta.........................................................................................................................................303
  src_ip......................................................................................................................................304
  src_mac..................................................................................................................................304
  src_port..................................................................................................................................304
  ttl...........................................................................................................................................304
  tunnel.....................................................................................................................................304
  vlan_id...................................................................................................................................304
doca_flow_action_descs_meta.................................................................................................... 304
  pkt_meta.................................................................................................................................305
  u32..........................................................................................................................................305
doca_flow_action_field.............................................................................................................. 305
  address..................................................................................................................................305
  offset.....................................................................................................................................305
doca_flow_actions......................................................................................................................305
action_idx...................................................................................................................................305
crypto_id...................................................................................................................................305
decap.........................................................................................................................................305
encap.........................................................................................................................................306
flags...........................................................................................................................................306
has_encap.................................................................................................................................306
meta...........................................................................................................................................306
mod_dst_ip................................................................................................................................306
mod_dst_mac............................................................................................................................306
mod_dst_port............................................................................................................................306
mod_src_ip................................................................................................................................306
mod_src_mac............................................................................................................................306
mod_src_port............................................................................................................................306
mod_vlan_id..............................................................................................................................307
proto_type..................................................................................................................................307
security...................................................................................................................................... 307
ttl................................................................................................................................................307
doca_flow_aged_query................................................................................................................. 307
user_data...................................................................................................................................307
doca_flow_cfg...............................................................................................................................307
cb...............................................................................................................................................307
mode_args.................................................................................................................................307
nr_shared_resources............................................................................................................... 307
queue_depth..............................................................................................................................308
queues.......................................................................................................................................308
resource....................................................................................................................................308
unbind_cb..................................................................................................................................308
doca_flow_encap_action.............................................................................................................. 308
dst_ip.........................................................................................................................................308
dst_mac.....................................................................................................................................308
src_ip.........................................................................................................................................308
src_mac.....................................................................................................................................308
tun..............................................................................................................................................309
vlan_tci.......................................................................................................................................309
doca_flow_error............................................................................................................................309
message.................................................................................................................................... 309
type............................................................................................................................................309
doca_flow_fwd..............................................................................................................................309
idx...............................................................................................................................................309
next_pipe.......................................................................................................................................309
num_of_queues................................................................................................................................309
ordered_list_pipe..........................................................................................................................309
pipe................................................................................................................................................310
port_id............................................................................................................................................310
rss_flags........................................................................................................................................310
rss_queues......................................................................................................................................310
shared_rss_id................................................................................................................................310
type.............................................................................................................................................310
doca_flow_grpcBindableObj..............................................................................................................310
pipe_id...........................................................................................................................................310
port_id...........................................................................................................................................310
type.............................................................................................................................................310
doca_flow_grpc_fwd........................................................................................................................311
fwd................................................................................................................................................311
next_pipe_id..................................................................................................................................311
doca_flow_grpc_pipe_cfg...............................................................................................................311
cfg...............................................................................................................................................311
port_id...........................................................................................................................................311
doca_flow_grpc_response..............................................................................................................311
aging_res.......................................................................................................................................311
entry_id.........................................................................................................................................311
entry_status.................................................................................................................................312
error............................................................................................................................................312
nb_entries_processed...................................................................................................................312
pipe_id...........................................................................................................................................312
success.......................................................................................................................................312
switch_port_id..............................................................................................................................312
doca_flow_ip_addr.........................................................................................................................312
ipv4_addr.....................................................................................................................................312
ipv6_addr.....................................................................................................................................312
type............................................................................................................................................313
doca_flow_match..........................................................................................................................313
flags............................................................................................................................................313
in_dst_ip.......................................................................................................................................313
in_dst_mac....................................................................................................................................313
in_dst_port..................................................................................................................................313
in_eth_type................................................................................................................................. 313
in_l4_type................................................................................................................................. 313
in_src_ip................................................................................................................................. 313
in_src_mac............................................................................................................................... 313
in_src_port............................................................................................................................... 313
in_tcp_flags............................................................................................................................ 314
in_vlan_tci............................................................................................................................... 314
meta.............................................................................................................................................. 314
out_dst_ip................................................................................................................................. 314
out_dst_mac............................................................................................................................... 314
out_dst_port............................................................................................................................... 314
out_eth_type............................................................................................................................. 314
out_l4_type............................................................................................................................... 314
out_src_ip................................................................................................................................. 314
out_src_mac............................................................................................................................... 314
out_src_port............................................................................................................................... 315
out_tcp_flags............................................................................................................................. 315
out_vlan_tci............................................................................................................................... 315
tun............................................................................................................................................... 315
doca_flow_meta......................................................................................................................... 315
align.............................................................................................................................................. 315
ipsec_syndrome........................................................................................................................ 315
lag_port....................................................................................................................................... 315
mark............................................................................................................................................. 315
nisp_syndrome.......................................................................................................................... 316
pkt_meta....................................................................................................................................... 316
port_meta..................................................................................................................................... 316
type............................................................................................................................................ 316
u32................................................................................................................................................ 316
zone............................................................................................................................................ 316
doca_flow_monitor..................................................................................................................... 316
aging............................................................................................................................................ 316
cbs............................................................................................................................................... 316
cir............................................................................................................................................... 316
flags............................................................................................................................................. 316
shared_counter_id...................................................................................................................... 317
shared_meter_id........................................................................................................................ 317
user_data..................................................................................................................................... 317
doca_flow_ordered_list................................................................................................................317
  elements..................................................................................................................................317
  idx........................................................................................................................................317
  size......................................................................................................................................317
doca_flow_pipe_attr..................................................................................................................317
  is_root.................................................................................................................................317
  name.................................................................................................................................318
  nb_actions.........................................................................................................................318
  nb_flows............................................................................................................................318
  nb_ordered_lists................................................................................................................318
  type.................................................................................................................................318
doca_flow_pipe_cfg..................................................................................................................318
  action_descs.......................................................................................................................318
  actions...............................................................................................................................318
  attr......................................................................................................................................318
  match...............................................................................................................................318
  match_mask.......................................................................................................................318
  monitor..............................................................................................................................319
  ordered_lists.....................................................................................................................319
  port.....................................................................................................................................319
doca_flow_port_cfg..................................................................................................................319
  devargs.............................................................................................................................319
  port_id...............................................................................................................................319
  priv_data_size...................................................................................................................319
  type....................................................................................................................................319
doca_flow_query.....................................................................................................................319
  total_bytes.........................................................................................................................319
  total_pkts..........................................................................................................................320
doca_flow_resource_crypto_cfg...............................................................................................320
  action_type.........................................................................................................................320
  fwd.....................................................................................................................................320
  header_type.......................................................................................................................320
  key.....................................................................................................................................320
  key_sz..................................................................................................................................320
  net_type............................................................................................................................320
  proto_type.........................................................................................................................321
  reformat_data.....................................................................................................................321
  reformat_data_sz................................................................................................................321
reformat_type............................................................................................................................321
security_ctx............................................................................................................................... 321
doca_flow_resource_meter_cfg.................................................................................................. 321
cbs..............................................................................................................................................321
cir............................................................................................................................................... 321
doca_flow_resource_rss_cfg....................................................................................................... 322
flags........................................................................................................................................... 322
nr_queues..................................................................................................................................322
queues_array.............................................................................................................................322
doca_flow_resources....................................................................................................................322
nb_counters...............................................................................................................................322
nb_meters...................................................................................................................................322
doca_flow_shared_resource_cfg.................................................................................................322
doca_flow_shared_resource_result............................................................................................ 323
doca_flow_tun...............................................................................................................................323
audp_hdr....................................................................................................................................323
esp_sn........................................................................................................................................323
esp_spi.......................................................................................................................................323
gre_key...................................................................................................................................... 323
gtp_teid......................................................................................................................................323
key_present............................................................................................................................... 323
nisp_hdr.....................................................................................................................................323
protocol.....................................................................................................................................323
type.......................................................................................................................................... 324
vxlan_tun_id.............................................................................................................................. 324
doca_ipsec_sa_antireplay............................................................................................................ 324
antireplay_enable......................................................................................................................324
esn_overlap_event_arm........................................................................................................... 324
hard_lifetime_arm....................................................................................................................324
remove_flow_enable.................................................................................................................324
remove_flow_packet_count......................................................................................................324
remove_flow_soft_lifetime....................................................................................................... 325
replay_win_state.......................................................................................................................325
replay_win_sz.......................................................................................................................... 325
soft_lifetime_arm......................................................................................................................325
doca_ipsec_sa_attrs.................................................................................................................... 325
antireplay...................................................................................................................................325
direction.................................................................................................................................... 325
Chapter 4. Data Fields.................................................................................................................336
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:

- App Shield
- App Shield Attributes
- arg parser
- Core
  - DOCA Buffer
  - DOCA Buffer Inventory
  - DOCA Context
  - DOCA Device
  - DOCA DPDK
  - DOCA Error
  - DOCA Hotplug
  - DOCA Memory Map
  - DOCA RDMA
  - DOCA Types
- Comm Channel
- Compatibility Management
- DOCA COMPRESS engine
- Environment Configurations
- ct
- DOCA DMA engine
- Deep packet inspection
- Remote deep packet inspection (grpc)
- flow net define
- Flow
- flow net define
- IPsec
2.1.  App Shield

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

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

Shadow function - get attribute value for a attestation.

**Parameters**

- **attestation**: single attestation handler
- **attr**: Attribute to get the info on the attestation

**Returns**

return the info requested, need to cast

**Description**

Do not use this function, recommended to use doca_apsh_attestation_info_get
const __DOCA_EXPERIMENTAL void *__doca_apsh_envar_info_get (doca_apsh_envar *envar, doca_apsh_envar_attr attr)
Shadow function - get attribute value for an environment variable.

Parameters
envar
    single envar handler

attr
    Attribute to get the info on the envar

Returns
return the info requested, need to cast

Description
Do not use this function, recommended to use doca_apsh_envar_info_get

const __DOCA_EXPERIMENTAL void *__doca_apsh_handle_info_get (doca_apsh_handle *handle, doca_apsh_handle_attr attr)
Shadow function - get attribute value for a handle.

Parameters
handle
    single handle handler

attr
    Attribute to get the info on the handle

Returns
return the info requested, need to cast

Description
Do not use this function, recommended to use doca_apsh_handle_info_get
const __DOCA_EXPERIMENTAL void __doca_apsh_ldrmodule_info_get (doca_apsh_ldrmodule *ldrmodule, doca_apsh_ldrmodule_attr attr)

Shadow function - get attribute value for a module.

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_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

doca_error_t __doca_apsh_sys_config
(doca_apsh_system *system,
doca_apsh_system_config_attr 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

```c
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

```c
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

__DOCA_EXPERIMENTAL void
doca_apsh_attestation_free (doca_apsh_attestation **attestation)
Destroys a attestation context.

Parameters
attestation
   Attestation opaque pointer of the process to destroy

doca_error_t doca_apsh_attestation_get
(doca_apsh_process *process,
const char *exec_hash_map_path,
doca_apsh_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_error_t doca_apsh_attst_refresh
doca_apsh_attestation attestication,
int *attestation_size)
refresh single attestation handler of a process with new snapshot

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

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if modules list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to modules array.
- DOCA_ERROR_NOT_FOUND - if process hasn’t been found.

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

__DOCA_EXPERIMENTAL doca_apsh_ctx
*doca_apsh_create (void)
Create a new apsh handler.

Returns
apsh context required for creating system handler, NULL on failure
Description
Allocate memory and init the opaque struct for apsh handler. Before using the system handler use doca_apsh_start

__DOCA_EXPERIMENTAL void doca_apsh_destroy (doca_apsh_ctx *ctx)
Free the APSH memory and close connections.

Parameters
ctx
apsh context to destroy

doca_error_t doca_apsh_dma_dev_set (doca_apsh_ctx *ctx, doca_dev *dma_dev)
Set apsh dma device.

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

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for dma_dev_name.

Description
This is a Mandatory setter
__DOCA_EXPERIMENTAL void
doca_apsh_envvars_free (doca_apsh_envvar **envvars)
Destroys a envvars context.

Parameters

envvars
Array of envvars opaque pointers of the process to destroy

doca_error_t doca_apsh_envvars_get
(doca_apsh_process *process,
doca_apsh_envarenvars, int *envars_size)
Get array of current process environment variables.

Parameters

process
Process handler

envvars
Array of environment variables opaque pointers of the process. in case process doesn’t have any envvars, 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 envvars list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to envvars 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_envvars_free to free it.

Note:
currently supported only for windows systems.
__DOCA_EXPERIMENTAL void
doca_apsh_handles_free (doca_apsh_handle
**handles)
Destroys a handles context.

Parameters
handles
   Array of handles opaque pointers of the process to destroy

doca_error_t doca_apsh_handles_get
(doca_apsh_process *process,
doca_apsh_handle *handles, 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:
__DOCA_EXPERIMENTAL void
doca_apsh_ldrmodules_free (doca_apsh_ldrmodule **ldrmodules)
Destroys a ldrmodules context.

Parameters

**ldrmodules**
Array of ldrmodules opaque pointers of the process to destroy

doca_error_t doca_apsh_ldrmodules_get
(doca_apsh_process *process,
doca_apsh_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_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.

```c
__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

```c
doca_error_t doca_apsh_modules_get
(doca_apsh_system *system,
doca_apsh_module modules, 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.

```c
__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

```c
doca_error_t doca_apsh_privileges_get
(doca_apsh_process *process,
doca_apsh_privilege*privileges, int *privileges_size)
```

Get array of current process privileges.

Parameters

`process`
Process handler

`privileges`
Array of privileges opaque pointers of the process. In case process doesn’t have any privileges, will return NULL.

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

Returns

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

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

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

Note:
currently supported only for windows systems.

__DOCA_EXPERIMENTAL void
doca_apsh_processes_free (doca_apsh_process **processes)
Destroys a process context.

Parameters
processes
Array of process opaque pointers of the systems to destroy

doca_error_t doca_apsh_processes_get
(doca_apsh_system *system,
doca_apsh_processprocesses, int *processes_size)
Get array of current processes running on the system.

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

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_INITIALIZATION - if processes list initialization failed.
- DOCA_ERROR_NO_MEMORY - if cannot alloc memory to processes array.
Description
This function is multithreaded compatible with different system context, meaning do not call this function simultaneously with the same system context. The return array is snapshot, this is not dynamic array, need to free it.

doca_error_t doca_apsh_regex_dev_set (doca_apsh_CTX *ctx, doca_dev *regex_dev)
Set apsh regex device.

Parameters
ctx
    apsh handler
regex_dev
doca device with the capabilities of regex

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description
This is not a Mandatory setter

Note:
currently supported only for windows systems.

doca_error_t doca_apsh_start (doca_apsh_CTX *ctx)
Start apsh handler.

Parameters
ctx
    App Shield handler

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
Description
Start apsh handler and init connection to devices. Need to set apsh params with setter functions before starting the system. Mandatory setters: doca_apsh_dma_dev_set. Other setters can be query automatically but will take time.

doca_error_t doca_apsh_sys_dev_set
(doca_apsh_system *system, doca_dev_rep *dev)
Set system device.

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

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
  ▶ DOCA_ERROR_BAD_STATE - if system was already started.

Description
This is a Mandatory setter

doca_error_t doca_apsh_sys_kpgd_file_set
(doca_apsh_system *system, const char *system_kpgd_file_path)
Set system kpgd file.

Parameters
system
  system handler
**system_kpgd_file_path**

**Returns**
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type had been received.
- DOCA_ERROR_BAD_STATE - if system was already started.

**Description**
This is not a must setter.

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

Set system allowed memory regions.

**Parameters**

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

**Returns**
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.
- DOCA_ERROR_BAD_STATE - if system was already started.

**Description**
This is a Mandatory setter.
**doca_error_t doca_apsh_sys_os_symbol_map_set**
(doca_apsh_system *system, const char *system_os_symbol_map_path)

Set system os symbol map.

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

**Returns**
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - if cannot alloc new buffer for system_os_symbol_map_path.
- DOCA_ERROR_BAD_STATE - if system was already started.

**Description**
This is a Mandatory setter

**doca_error_t doca_apsh_sys_os_type_set**
(doca_apsh_system *system, doca_apsh_system_os os_type)

Set system os type.

**Parameters**
- **system**
  system handler
- **os_type**
  system os type - windows/linux

**Returns**
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_SUPPORTED - if unsupported OS type had been received.
- DOCA_ERROR_BAD_STATE - if system was already started.

**Description**

This is a must setter

```c
__DOCA_EXPERIMENTAL doca_apsh_system
*doca_apsh_system_create (doca_apsh_ctx *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`

```c
__DOCA_EXPERIMENTAL void
doca_apsh_system_destroy (doca_apsh_system *system)
```

Destroy system handler.

**Parameters**

- `system` - system context to destroy

**Description**

This will not destroy process/module/libs ...
doca_error_t doca_apsh_system_start
doca_apsh_system *system)
Start system handler.

Parameters

system
system handler

Returns

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

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

Description

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

__DOCA_EXPERIMENTAL void
doca_apsh_threads_free (doca_apsh_thread
**threads)
Destroys a threads context.

Parameters

threads
Array of threads opaque pointers of the process to destroy

doca_error_t doca_apsh_threads_get
(doca_apsh_process *process,
doca_apsh_threadthreads, int *threads_size)
Get array of current process threads.

Parameters

process
Process handler
**threads**
Array of threads opaque pointers of the process. In case process doesn’t have any threads, will return NULL.

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

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

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

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

```c
__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

```c
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.

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

Get attribute value for an 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

```c
#define doca_apsh_envar_info_get ((attr##_TYPE)__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)__doca_apsh_handle_info_get(handle, attr))
Get attribute value for a handle.
Get the requested info from handle handler. The info is right to the snapshot (at the get function moment) full list (type and descriptions) can be found in doca_apsh_attr.h

#define doca_apsh_ldrmodule_info_get
((attr##_TYPE)__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 (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 (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_privilege_info_get
((attr##_TYPE)__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

```c
#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

```c
#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 ...

```c
#define doca_apsh_thread_info_get ((attr##_TYPE)__doca_apsh_thread_info_get(thread, attr))
```
Get attribute value for a thread.

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

```c
#define doca_apsh_vad_info_get ((attr##_TYPE) (uintptr_t)__doca_apsh_vad_info_get(vad, attr))
```
Get attribute value for a vad.

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

### 2.2. App Shield Attributes

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

doca app shield attestation attributes

Values

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

enum doca_apsh_envar_attr

doca app shield envars attributes

Values

DOCA_APSH_ENVARS_PID
    envars pid
DOCA_APSH_ENVARS_COMM
    envars process name
DOCA_APSH_ENVARS_VARIABLE
    envars variable
DOCA_APSH_ENVARS_VALUE
    envars value
DOCA_APSH_ENVARS_WINDOWS_BLOCK = 1000
    envars windows environment block address
enum doca_apsh_handle_attr
doca app shield handle attributes

Values

DOCA_APSH_HANDLE_PID
handle process id

DOCA_APSH_HANDLE_COMM
handle process name

DOCA_APSH_HANDLE_VALUE
handle value

DOCA_APSH_HANDLE_TABLE_ENTRY
handle table entry

DOCA_APSH_HANDLE_TYPE
handle type

DOCA_APSH_HANDLE_ACCESS
handle access

DOCA_APSH_HANDLE_NAME
handle name

enum doca_apsh_ldrmodule_attr
doca app shield LDR-Modules attributes

Values

DOCA_APSH_LDRMODULE_PID
ldrmodule process pid

DOCA_APSH_LDRMODULE_COMM
ldrmodule process name

DOCA_APSH_LDRMODULE_BASE_ADDRESS
ldrmodule base address

DOCA_APSH_LDRMODULE_LIBRARY_PATH
ldrmodule loaded library path

DOCA_APSH_LDRMODULE_WINDOWS_BASE_DLL_NAME = 1000
ldrmodule full dll name

DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE
ldrmodule size of image

DOCA_APSH_LDRMODULE_WINDOWS_INLOAD
ldrmodule appear in inload list

DOCA_APSH_LDRMODULE_WINDOWS_INMEM
ldrmodule appear in inmem list

DOCA_APSH_LDRMODULE_WINDOWS_ININIT
ldrmodule appear in ininit list
enum doca_apsh_lib_attr
  doca app shield lib attributes

Values
  DOCA_APSH_LIB_PID
    lib pid
  DOCA_APSH_LIB_COMM
    lib name
  DOCA_APSH_LIB_LIBRARY_PATH
    lib loaded library path
  DOCA_APSH_LIB_WINDOWS_FULL_DLL_NAME = 1000
    lib full dll name
  DOCA_APSH_LIB_WINDOWS_SIZE_OFIMAGE
    lib size of image
  DOCA_APSH_LIB_LINUX_LOAD_ADRESS = 2000
    lib load address

enum doca_apsh_module_attr
  doca app shield module attributes

Values
  DOCA_APSH_MODULES_OFFSET
    module offset
  DOCA_APSH_MODULES_NAME
    module name
  DOCA_APSH_MODULES_SIZE
    module size

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

Values
  DOCA_APSH_PRIVILEGES_PID
    privilege process pid
  DOCA_APSH_PRIVILEGES_COMM
    privilege process name
  DOCA_APSH_PRIVILEGES_NAME
    privilege name, for example: SeTcbPrivilege
  DOCA_APSH_PRIVILEGES_IS_ON
is the privilege turned on or off. For Windows this is the outcome of get(PRESENT) &&
(get(ENABLED) || get(DEFAULT))

**DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT** = 1000
privilege present flag

**DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED**
privilege enabled flag

**DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT**
privilege enabled by default flag

**enum doca_apsh_process_attr**
doca app shield process attributes

**Values**

**DOCA_APSH_PROCESS_PID**
process id

**DOCA_APSH_PROCESS_PPID**
process parent id

**DOCA_APSH_PROCESS_COMM**
process executable name

**DOCA_APSH_PROCESS_STATE**
process state

**DOCA_APSH_PROCESS_CPU_TIME**
process cpu time [ps]

**DOCA_APSH_PROCESSINDOWS_OFFSET** = 1000
process offset

**DOCA_APSH_PROCESS_WINDOWS_THREADS**
process thread count

**DOCA_APSH_PROCESS_LINUX_GID** = 2000
process group id

**DOCA_APSH_PROCESS_LINUX_UID**
process user id

**enum doca_apsh_system_config_attr**
doca app shield configuration attributes

**Values**

**DOCA_APSH_OS_SYMBOL_MAP**
os symbol map path

**DOCA_APSH_MEM_REGION**
memory region path

**DOCA_APSH_KPGD_FILE**
kpgd file path
enum doca_apsh_system_os
system os types

Values

DOCA_APSH_SYSTEM_LINUX
linux
DOCA_APSH_SYSTEM_WINDOWS
windows

enum doca_apsh_thread_attr
doca app shield thread attributes

Values

DOCA_APSH_THREAD_PID
thread process id
DOCA_APSH_THREAD_TID
thread id
DOCA_APSH_THREAD_STATE
thread state

DOCA_APSH_THREAD_WINDOWS_WAIT_REASON = 1000
thread wait reason

DOCA_APSH_THREAD_WINDOWS_OFFSET
thread offset

DOCA_APSH_THREAD_LINUX_PROC_NAME = 2000
thread process name

DOCA_APSH_THREAD_LINUX_THREAD_NAME
thread name

enum doca_apsh_vad_attr
doca app shield virtual address descriptor attributes

Values

DOCA_APSH_VMA_PID
vma process id

DOCA_APSH_VMA_OFFSET
vma offset

DOCA_APSH_VMA_PROTECTION
vma protection

DOCA_APSH_VMA_VM_START
vma vm start

DOCA_APSH_VMA_VM_END
vma vm end

DOCA_APSH_VMA_PROCESS_NAME
vma process name

DOCA_APSH_VMA_FILE_PATH
vma file path

DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE = 1000
vma commit charge

DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY
vma private memory

typedef char
*DOCA_APSH_ATTESTATION_COMM_TYPE

attestation comm type

typedef uint64_t
DOCA_APSH_ATTESTATION_END_ADDRESS_TYPE

attestation end address type
typedef bool
DOCA_APSH_ATTESTATION_HASH_DATA_IS_PRESENT_TYPE
attestation hash data is present type

typedef int
DOCA_APSH_ATTESTATION_MATCHING_HASHES_TYPE
attestation matching hashes type

typedef int
DOCA_APSH_ATTESTATION_PAGES_NUMBER_TYPE
attestation pages number type

typedef int
DOCA_APSH_ATTESTATION_PAGES_PRESENT_TYPE
attestation pages present type

typedef char
*DOCA_APSH_ATTESTATION_PATH_OF_MEMORY_AREA_TYPE
attestation path of memory area type

typedef unsigned int
DOCA_APSH_ATTESTATION_PID_TYPE
attestation pid type

typedef char
*DOCA_APSH_ATTESTATION_PROTECTION_TYPE
attestation protection type

typedef uint64_t
DOCA_APSH_ATTESTATION_START_ADDRESS_TYPE
attestation start address type

typedef doca_dev *
DOCA_APSH_DMA_DEV_TYPE
dma dev name
typedef char *DOCA_APSH_ENVARS_COMM_TYPE
envars comm type

typedef unsigned int DOCA_APSH_ENVARS_PID_TYPE
envars pid type

typedef char *DOCA_APSH_ENVARS_VALUE_TYPE
envars value type

typedef char *DOCA_APSH_ENVARS_VARIABLE_TYPE
envars variable type

typedef uint64_t
DOCA_APSH_ENVARS_WINDOWS_BLOCK_TYPE
envars windows block address type

typedef uint64_t
DOCA_APSH_HANDLE_ACCESS_TYPE
handle access type

typedef char *DOCA_APSH_HANDLE_COMM_TYPE
handle comm type

typedef char *DOCA_APSH_HANDLE_NAME_TYPE
handle name type

typedef unsigned int
DOCA_APSH_HANDLE_PID_TYPE
handle pid type

typedef uint64_t
DOCA_APSH_HANDLE_TABLE_ENTRY_TYPE
handle table entry type

typedef char *DOCA_APSH_HANDLE_TYPE_TYPE
handle type type
typedef uint64_t DOCA_APSH_HANDLE_VALUE_TYPE
handle value type

typedef int DOCA_APSH_HASHTEST_LIMIT_TYPE
limit of vm areas to attest

typedef char *DOCA_APSH_KPGD_FILE_TYPE
kpgd file path

typedef uint64_t
DOCA_APSH_LDRMODULE_BASE_ADDRESS_TYPE
ldrmodule base adress type

typedef char
*DOCA_APSH_LDRMODULE_COMM_TYPE
ldrmodule comm type

typedef char
*DOCA_APSH_LDRMODULE_LIBRARY_PATH_TYPE
ldrmodule library path type

typedef unsigned int
DOCA_APSH_LDRMODULE_PID_TYPE
ldrmodule pid type

typedef char
*DOCA_APSH_LDRMODULE_WINDOWS_BASE_DLL_NAME_TYPE
ldrmodule windows BASE dll name type

typedef bool
DOCA_APSH_LDRMODULE_WINDOWS_ININIT_TYPE
ldrmodule ininit type
typedef bool
DOCA_APSH_LDRMODULE_WINDOWS_INLOAD_TYPE
ldrmodule inload type

typedef bool
DOCA_APSH_LDRMODULE_WINDOWS_INMEM_TYPE
ldrmodule inmem type

typedef unsigned long
DOCA_APSH_LDRMODULE_WINDOWS_SIZE_OF_IMAGE_TYPE
ldrmodule size of image type

typedef char *
DOCA_APSH_LIB_COMM_TYPE
lib comm type

typedef char *
DOCA_APSH_LIB_LIBRARY_PATH_TYPE
lib loaded library path type

typedef uint64_t
DOCA_APSH_LIB_LINUX_LOAD_ADRESS_TYPE
lib load adress type

typedef unsigned int
DOCA_APSH_LIB_PID_TYPE
lib pid type

typedef char *
DOCA_APSH_LIB_WINDOWS_FULL_DLL_NAME_TYPE
lib full dll name type

typedef unsigned long
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_OS_SYMBOL_MAP_TYPE
os symbol map path

typedef DOCA_APSH_OS_TYPE_TYPE
os type

typedef char *
*DOCA_APSH_PRIVILEGES_COMM_TYPE
privilege process name

typedef bool DOCA_APSH_PRIVILEGES_IS_ON_TYPE
privilege is on type

typedef char *DOCA_APSH_PRIVILEGES_NAME_TYPE
privilege name type
typedef unsigned int
DOCA_APSH_PRIVILEGES_PID_TYPE
privilege process pid

typedef bool
DOCA_APSH_PRIVILEGES_WINDOWS_DEFAULT_TYPE
privilege windows enabled by default type

typedef bool
DOCA_APSH_PRIVILEGES_WINDOWS_ENABLED_TYPE
privilege windows enabled type

typedef bool
DOCA_APSH_PRIVILEGES_WINDOWS_PRESENT_TYPE
privilege windows present type

typedef char *
DOCA_APSH_PROCESS_COMM_TYPE
process comm type

typedef uint64_t
DOCA_APSH_PROCESS_CPU_TIME_TYPE
process cpu time type

typedef int
DOCA_APSH_PROCESS_LIMIT_TYPE
limit of processes number

typedef unsigned int
DOCA_APSH_PROCESS_LINUX_GID_TYPE
process gid type

typedef unsigned int
DOCA_APSH_PROCESS_LINUX_UID_TYPE
process uid type
typedef unsigned int DOCA_APSH_PROCESS_PID_TYPE
  process pid type

typedef unsigned int DOCA_APSH_PROCESS_PPID_TYPE
  process pid type

typedef long DOCA_APSH_PROCESS_STATE_TYPE
  process state type

typedef uint64_t DOCA_APSH_PROCESS_WINDOWS_OFFSET_TYPE
  process offset type

typedef int DOCA_APSH_PROCESS_WINDOWS_THREADS_TYPE
  process threads type

typedef char *DOCA_APSH_REGEX_DEV_TYPE
  regex dev name

typedef int DOCA_APSH_STRING_LIMIT_TYPE
  length limit of apsh_read_str

typedef char *DOCA_APSH_THREAD_LINUX_PROC_NAME_TYPE
  thread proc name type

typedef char *DOCA_APSH_THREAD_LINUX_THREAD_NAME_TYPE
  thread thread name type
typedef unsigned int DOCA_APSH_THREAD_PID_TYPE
thread pid type

typedef long DOCA_APSH_THREAD_STATE_TYPE
thread state type

typedef unsigned int DOCA_APSH_THREAD_TID_TYPE
thread tid type

typedef uint64_t DOCA_APSH_THREAD_WINDOWS_OFFSET_TYPE
thread offset type

typedef unsigned char DOCA_APSH_THREAD_WINDOWS_WAIT_REASON_TYPE
thread wait reason type

typedef int DOCA_APSH_THREADS_LIMIT_TYPE
limit of threads number

typedef int DOCA_APSH_VADS_LIMIT_TYPE
limit of vads number

typedef doca_dev_rep *DOCA_APSH_VHCA_ID_TYPE
vhca id

typedef char *DOCA_APSH_VMA_FILE_PATH_TYPE
vma file path type

typedef uint64_t DOCA_APSH_VMA_OFFSET_TYPE
vma offset type

typedef unsigned int DOCA_APSH_VMA_PID_TYPE
vma pid type
typedef char
*DOCA_APSH_VMA_PROCESS_NAME_TYPE
vma file path type

typedef char *
*DOCA_APSH_VMA_PROTECTION_TYPE
vma protection type

typedef uint64_t
DOCA_APSH_VMA_VM_END_TYPE
vma vm end type

typedef uint64_t
DOCA_APSH_VMA_VM_START_TYPE
vma vm start type

typedef int
DOCA_APSH_VMA_WINDOWS_COMMIT_CHARGE_TYPE
vma commit charge type

typedef int
DOCA_APSH_VMA_WINDOWS_PRIVATE_MEMORY_TYPE
vma private memory type

typedef int
DOCA_APSH_WINDOWS_ENVARSLIMIT_TYPE
length limit of envars for windows

2.4. Core

DOCA Buffer

DOCA Buffer Inventory

DOCA Context

DOCA Device
DOCA Buffer Core

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

doca_error_t doca_buf_get_data (doca_buf *buf, void **data)
Get the buffer’s data.

Parameters

buf
  DOCA Buf element.

data
  The data of the buffer.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
docta_error_t doca_buf_get_data_len (doca_buf *buf, size_t *data_len)
Get buffer’s data length.

Parameters
buf
   DOCA Buf element.
data_len
   The data length of the buffer.

Returns
DOCA_SUCCESS - in case of success. docta_error code - in case of failure:
   • DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

docta_error_t doca_buf_get_head (doca_buf *buf, void **head)
Get the buffer’s head.

Parameters
buf
   DOCA Buf element.
head
   The head of the buffer.

Returns
DOCA_SUCCESS - in case of success. docta_error code - in case of failure:
   • DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

docta_error_t doca_buf_get_len (doca_buf *buf, size_t *len)
Get the buffer’s length.

Parameters
buf
   DOCA Buf element.
len
   The length of the buffer.

Returns
DOCA_SUCCESS - in case of success. docta_error code - in case of failure:
DOCA_ERRORINVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_get_lkey (const doca_buf *buf, doca_dev *dev, uint32_t doca_access_flags, uint32_t *lkey)
Get lkey with doca_access_flags access for a DOCA buffer of a DOCA device.

Parameters
buf
The DOCA buffer to get lkey for.
dev
The DOCA device to get lkey for.
doca_access_flags
LKey access flags [see enum doca_access_flags].
lkey
The returned LKey.

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

- DOCA_ERRORINVALID_VALUE - if an invalid input had been received or if cannot find mkey by the given device.
- DOCA_ERRORNOT_SUPPORTED - if the given access flags is not supported

doca_error_t doca_buf_get_refcount (doca_buf *buf, uint16_t *refcount)
Get the reference count of the object.

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

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

- DOCA_ERRORINVALID_VALUE - if an invalid input had been received.
doxa_error_t doca_buf_list_chain (doca_buf *list1, doca_buf *list2)
Append list2 to list1.

Parameters

list1
DOCA Buf representing list1. Buf must be the head of the list and have the linked list extension.

list2
DOCA Buf representing list2. Buf must be the head of the list and have the linked list extension.

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 one of the buffers is not a head of a list or the 'Linked List' extension of one of the buffers is disabled.

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

doxa_error_t doca_buf_list_is_first (const doca_buf *buf, bool *is_first)
Check if provided DOCA Buf is the first element in a linked list.

Parameters

buf
DOCA Buf element. Buf must have the linked list extension.

is_first
If 'Linked List' extension is enabled: true if buf is the first element, false if it is not If 'Linked List' extension is disabled: true

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_list_is_last (const doca_buf *buf, bool *is_last)
Check if provided DOCA Buf is the last element in a linked list.

Parameters
buf
DOCA Buf element. Buf must have the linked list extension.
is_last
If 'Linked List' extension is enabled: true if buf is the last element, false if it is not If 'Linked List' extension is disabled: true

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_list_last (doca_buf *buf, doca_buf **last_buf)
Get last DOCA Buf in linked list.

Parameters
buf
DOCA Buf element. Buf must have the linked list extension.
last_buf
The last DOCA Buf in the linked list, which may be buf, or buf if the 'Linked List' extension of the buffer is disabled.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_buf_list_next (doca_buf *buf, doca_buf **next_buf)
Get next DOCA Buf in linked list.

Parameters
buf
DOCA Buf element. Buf must have the linked list extension.
**next_buf**

The next DOCA Buf or null if the ‘Linked List’ extension of the buffer is disabled.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

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

Get the number of the elements in list.

**Parameters**

- **buf**
  
  DOCA Buf element. Buf must be a head of a list and have the linked list extension.

- **num_elements**
  
  If ‘Linked List’ extension is enabled: Number of elements in list If ‘Linked List’ extension is disabled: 1

**Returns**

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

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

**doca_error_t doca_buf_list_unchain (doca_buf *list1, doca_buf *list2)**

Separate list2 from list1.

**Parameters**

- **list1**
  
  DOCA Buf representing list1. Must have the linked list extension.

- **list2**
  
  DOCA Buf representing list2, list2 should be contained in list1. list2 must be different from list1 and must have the linked list extension.

**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 ‘Linked List’ extension of one of the buffers is disabled.
**Description**

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

After: +----+ +----+ +----+ +----+ +----+ list1 -> |1 |->|2 |->|3 |->|4 |->|5 | +----+ +----+ +----+ +----+ +----+

**docta_error_t docta_buf_refcount_add (docta_buf *buf, uint16_t *refcount)**

Increase the object reference count by 1.

**Parameters**

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

**Returns**

- **DOCA_ERROR_NOT_SUPPORTED**

**Description**

- **Note**: This function is not supported yet.

**docta_error_t docta_buf_refcount_rm (docta_buf *buf, uint16_t *refcount)**

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

**Parameters**

- **buf**: DOCA Buf element.
- **refcount**: The number of references to the object before this operation took place.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description
When refcnt 0 reached, all related resources should be released. For example if the element points into some mmap its state will be adjusted accordingly. If DOCA_BUF_EXTENSION_LINKED_LIST is selected the buf must be the head of a list

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

Parameters
- **buf**
  DOCA Buf element.
- **data**
  Data address.
- **data_len**
  Data length.

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or if data address and length are outside buffer’s limits.

Description
Set data pointer and data length

```
| __data_len__    | /     |
```

```
+-----------+-----+-----------------+ Before | |data | | +---------+-----------------+ +-----------+-----+-----------------+ After | |data | | +---------+-----------------+
```

2.4.2. DOCA Buffer Inventory

Core

The DOCA buffer inventory manages a pool of doca_buf objects. Each buffer obtained from an inventory is a descriptor that points to a memory region from a doca_mmap memory range of the user’s choice.
doeca_error_t doeca_buf_inventory_buf_by_addr
(doca_buf_inventory *inventory, doca_mmap *mmap, void
*addr, size_t len, doca_buf **buf)
Allocate single element from buffer inventory and point it to the buffer defined by ‘addr’ & ‘len’ arguments.

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

Returns
DOCA_SUCCESS - in case of success. doeca_error code - in case of failure:
    ▶ DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or if there is no suitable memory range for the given address and length.
    ▶ DOCA_ERROR_NOT_PERMITTED - if doca_mmap or doca_buf_inventory is un-started/stopped.
    ▶ DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

doeca_error_t doeca_buf_inventory_buf_by_args
(doca_buf_inventory *inventory, doca_mmap *mmap, void
*addr, size_t len, void *data, size_t data_len, doca_buf
**buf)
Allocate single element from buffer inventory and point it to the buffer defined by ‘addr’, ‘len’, ‘data’ and ‘data_len’ arguments.

Parameters
inventory
    The DOCA Buf inventory.
mmap
    DOCA memory map structure.
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.

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or if there is no
  suitable memory range for the given address and length or if data address and length are
  outside buffer’s limits.
- DOCA_ERROR_NOT_PERMITTED - if doca_mmap or doca_buf_inventory is un-started/
  stopped.
- DOCA_ERROR_NO_MEMORY - if doca_buf_inventory is empty.

doca_error_t doca_buf_inventory_buf_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.

mmap
DOCA memory map structure.

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.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received or if there is no suitable memory range for the given address and length.

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

- **DOCA_ERROR_NO_MEMORY** - if doca_buf_inventory is empty.

```c
#include <doca.h>

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

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

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

**Returns**
- **DOCA_SUCCESS** - in case of success. doca_error code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received.
  - **DOCA_ERROR_NOT_PERMITTED** - if src_buf mmap or input inventory unstarted/stopped or src_buf inventory extensions and the input inventory extensions are incompatible.
  - **DOCA_ERROR_NO_MEMORY** - if cannot alloc new doca_buf from the given inventory.

```c
#include <doca.h>

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

Allocates buffer inventory with default/unset attributes.

**Parameters**
- **user_data**
- **num_elements**
  - Initial number of elements in the inventory.
- **extensions**
  - Bitmap of extensions enabled for the inventory described in doca_buf.h.
- **buf_inventory**
  - Buffer inventory with default/unset attributes.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_buf_inventory.

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

doca_error_t doca_buf_inventory_destroy
doca_buf_inventory *inventory)
Destroy buffer inventory structure.

Parameters
inventory
Buffer inventory structure.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if not all allocated elements had been returned to the inventory.

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

doca_error_t doca_buf_inventory_get_list_supported
doca_buf_inventory *inventory, uint8_t *list_supported)
Check if DOCA buffer inventory supports list of DOCA buffers.

Parameters
inventory
A given DOCA buffer inventory.
list_supported
Indicating whether DOCA list buffer is supported (1 means supported, 0 means not supported).
Returns
DOCA_SUCCESS - In case of success. Error code - on failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
see enum doca_buf_extension for more details.

doca_error_t doca_buf_inventory_get_num_elements
(doca_buf_inventory *inventory, uint32_t *num_of_elements)
Read the total number of elements in a DOCA Inventory.

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

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
The total number of elements type: uint32_t.

doca_error_t doca_buf_inventory_get_num_free_elements
(doca_buf_inventory *inventory, uint32_t *num_of_free_elements)
Get the total number of free elements in a DOCA Inventory.

Parameters
  inventory
    The DOCA Buf inventory.
  num_of_free_elements
    The total number of free elements in inventory.
Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

The total number of free elements type: uint32_t.

doca_error_t doca_buf_inventory_get_user_data (doca_buf_inventory *inventory, doca_data *user_data)

Get the user_data of a DOCA Inventory.

Parameters

inventory
The DOCA Buf inventory.

user_data
The user_data of inventory.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

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

doca_error_t doca_buf_inventory_start (doca_buf_inventory *inventory)

Start element retrieval from inventory.

Parameters

inventory
Buffer inventory structure.

Returns

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
Description
Un-started/stopped buffer inventory rejects all attempts to retrieve element. On first start verifies & finalizes the buffer inventory object configuration.

The following become possible only after start:

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

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

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

```
doca_error_t doca_buf_inventory_stop (doca_buf_inventory *inventory)
```

Stop element retrieval from inventory.

Parameters

- `inventory` - Buffer inventory structure.

Returns

- `DOCA_SUCCESS` - in case of success.
- `doca_error` code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - if an invalid input had been received.

Description

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

2.4.3. DOCA Context

Core

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

struct doca_job
Job structure describes request arguments for service provided by context.

doca_error_t doca_ctx_dev_add (doca_ctx *ctx, doca_dev *dev)
Add a device to a DOCA CTX.

Parameters
ctx
The CTX to add the device to.

dev
The device to add.

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is started.
- DOCA_ERROR_IN_USE - the device was already added.
- DOCA_ERROR_NOT_SUPPORTED - the provided device is not supported by CTX, i.e., the device is not useful for any job, or missing the capabilities.

doca_error_t doca_ctx_dev_rm (doca_ctx *ctx, doca_dev *dev)
Remove a device from a context.

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

dev
The device to remove.

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

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

**docta_error_t doca_ctx_get_event_driven_supported (docta_ctx *ctx, uint8_t *event_supported)**

Check if CTX supports event driven mode.

**Parameters**

- **ctx**
  - The library instance containing the WorkQ.

- **event_supported**
  - Boolean indicating whether event driven mode is supported.

**Returns**

- DOCA_SUCCESS - in case of success. Error code - on failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.

**Description**

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

**docta_error_t doca_ctx_get_max_num_ctx (uint32_t *max_num_ctx)**

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

**Parameters**

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

**Returns**

- DOCA_SUCCESS - in case max_num_ctx received the required value properly. Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - max_num_ctx is NULL.
doca_error_t doca_ctx_start (doca_ctx *ctx)
Finalizes all configurations, and starts the DOCA CTX.

Parameters
ctx
The DOCA context to start.

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

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

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

The following become possible only after start:

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

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

- Adding device to CTX using doca_ctx_dev_add()!
- Removing device from CTX using doca_ctx_dev_rm()!

doca_error_t doca_ctx_stop (doca_ctx *ctx)
Stop the context allowing reconfiguration.

Parameters
ctx
The DOCA context to stop.

Returns
DOCA_SUCCESS - In case of success. Error code - In case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_PERMITTED - either some jobs are still pending or not all WorkQs have been removed.

Description

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

doca_error_t doca_ctx_workq_add (doca_ctx *ctx,
doca_workq *workq)
Add a workQ to a context.

Parameters

ctx
  The library instance that will handle the jobs.

workq
  The WorkQ where you want to receive job completions.

Returns

DOCA_SUCCESS - In case of success. Error code - on failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is not started.
- DOCA_ERROR_IN_USE - same WorkQ already added.
- DOCA_ERROR_NO_MEMORY - memory allocation failed.
- DOCA_ERROR_INITIALIZATION - initialization of WorkQ failed.

Description

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

doca_error_t doca_ctx_workq_rm (doca_ctx *ctx,
doca_workq *workq)
Remove a DOCA WorkQ from a DOCA CTX.

Parameters

ctx
  The library instance containing the WorkQ.
The WorkQ to remove.

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - CTX is not started.
- DOCA_ERROR_NOT_PERMITTED - some jobs are still pending completion.
- DOCA_ERROR_NOT_FOUND - WorkQ does not exist within CTX.

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

doca_error_t doca_workq_create (uint32_t depth, doca_workq **workq)
Creates empty DOCA WorkQ object with default attributes.

Parameters
- depth
  The maximum number of inflight jobs.
- workq
  The newly created WorkQ.

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

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

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

doca_error_t doca_workq_destroy (doca_workq *workq)
Destroy a DOCA WorkQ.

Parameters
- workq
  The WorkQ to destroy.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - invalid input received.
- DOCA_ERROR_IN_USE - WorkQ not removed from one of the doca_ctx.

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

doca_error_t doca_workq_event_handle_arm (doca_workq *workq)
Arm the WorkQ to receive next completion event.

Parameters
workq
The WorkQ object to arm.

Returns

- DOCA_SUCCESS - workq has been successfully armed, event handle can be used to wait on events.
- DOCA_ERROR_BAD_STATE - event driven mode is not enabled. Try doca_workq_set_event_driven_enable().

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

doca_error_t doca_workq_event_handle_clear (doca_workq *workq, doca_event_handle_t handle)
Clear triggered events.

Parameters
workq
The WorkQ object that received the events.
Method used for clearing of events, this method should be called after an event has been received using the event handle. After this is called, the events will no longer be triggered, and the handle can be armed again. See \texttt{doca\_workq\_event\_handle\_arm()} for entire flow.

\textbf{Returns}

- \texttt{DOCA\_SUCCESS} - on success fully clearing triggered events.
- \texttt{DOCA\_ERROR\_BAD\_STATE} - event driven mode is not enabled. Try \texttt{doca\_workq\_set\_event\_driven\_enable()}.
- \texttt{DOCA\_ERROR\_OPERATING\_SYSTEM} - a system call has failed.

\textbf{doca\_error\_t doca\_workq\_get\_depth (const doca\_workq \*workq, uint32\_t \*depth)}

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

\textbf{Parameters}

- \texttt{workq}  
  The DOCA WorkQ.

- \texttt{depth}  
  The maximum number of inflight jobs allowed for workq.

\textbf{Returns}

- \texttt{DOCA\_SUCCESS} - in case of success. Error code - in case of failure:
- \texttt{DOCA\_ERROR\_INVALID\_VALUE} - received invalid input.

\textbf{doca\_error\_t doca\_workq\_get\_event\_driven\_enable (doca\_workq \*workq, uint8\_t \*enabled)}

Check if WorkQ event-driven mode is enabled.

\textbf{Parameters}

- \texttt{workq}  
  The WorkQ to query.

- \texttt{enabled}  
  0 or 1 indicating if event-driven mode is enabled.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

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

`doca_error_t doca_workq_get_event_handle (doca_workq *workq, doca_event_handle_t *handle)`
Get the event handle for waiting on events.

Parameters
- **workq**
  The WorkQ to query.
- **handle**
  The event handle of the WorkQ.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
  ▶ DOCA_ERROR_BAD_STATE - event driven mode is not enabled. Try `doca_workq_set_event_driven_enable()`.

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

`doca_error_t doca_workq_progress_retrieve (doca_workq *workq, doca_event *ev, int flags)`
Progress & retrieve single pending event.

Parameters
- **workq**
  The WorkQ object to poll for events.
- **ev**
  Event structure to be filled in case an event was received.
flags
Flags for progress/retrieval operations. A combination of enum doca_workq_retrieve_flags.

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

Description
Polling method for progress of submitted jobs and retrieval of events.
NOTE: for V1 retrieve supported for single event only.

doca_error_t doca_workq_set_depth (doca_workq *workq, uint32_t depth)
Set the maximum number of inflight jobs allowed for a DOCA WorkQ to a given value.

Parameters
- **workq**
  The DOCA WorkQ.
- **depth**
  The new maximum number of inflight jobs allowed for workq.

Returns
**DOCA_SUCCESS** - in case of success. Error code - in case of failure:
- **DOCA_ERROR_INVALID_VALUE** - received invalid input.

doca_error_t doca_workq_set_event_driven_enable (doca_workq *workq, uint8_t enable)
Enable WorkQ event-driven mode.

Parameters
- **workq**
  The WorkQ to query.
- **enable**
  0 or 1 indicating whether to enable event-driven mode or not.
Returns
DOCA_SUCCESS - in case event driven mode has been set, or is already set to same value. Error code - in case of failure:

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

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

doca_error_t doca_workq_submit (doca_workq *workq, doca_job *job)
Submit a job to a DOCA WorkQ.

Parameters

workq
The DOCA WorkQ used for progress and retrieval of jobs.

job
The job to submit, the job must be compatible with the WorkQ.

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_BAD_STATE - in case job->ctx is stopped.
- DOCA_ERROR_NO_MEMORY - in case the queue is full. See WorkQ depth.
- DOCA_ERROR_NOT_FOUND - in case the ctx is not associated to the workQ.

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

#define DOCA_ACTION_SDK_RANGE 16
Power 2 single SDK/context action type range.
2.4.4. DOCA Device

Core

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

enum doca_dev_rep_filter

Representor device filter by flavor

Multiple options possible but some are mutually exclusive.

Values

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

__DOCA_EXPERIMENTAL doca_devinfo

*doca_dev_as_devinfo (const doca_dev *dev)*

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

Parameters

dev

    The doca device instance.

Returns

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

**doaca_error_t doaca_dev_close (doaca_dev *dev)**

Destroy allocated local device instance.

Parameters

dev

    The local doca device instance.

Returns

DOCA_SUCCESS - in case of success.
DOCA_ERROR_IN_USE - failed to deallocate device resources.

doca_error_t doca_dev_open (doca_devinfo *devinfo, doca_dev **dev)
Initialize local device for use.

Parameters

  devinfo
  The devinfo structure of the requested device.

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

Returns

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

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

Description

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

__DOCA_EXPERIMENTAL doca_devinfo_rep*
doca_dev_rep_as_devinfo (doca_dev_rep *dev_rep)
Get representor device info from device. This should be useful when wanting to query information about device after opening it, and destroying the devinfo list.

Parameters

  dev_rep
  The representor doca device instance.

Returns

The matching doca_devinfo_rep instance in case of success, NULL in case dev_rep is invalid.
doça_error_t doça_dev_rep_close (doça_dev_rep *dev)
Destroy allocated representor device instance.

Parameters

**dev**
The representor doça 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.

doça_error_t doça_dev_rep_open (doça_devinfo_rep *devinfo, doça_dev_rep **dev_rep)
Initialize representor device for use.

Parameters

**devinfo**
The devinfo structure of the requested device.

**dev_rep**
Initialized representor doça 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.

doça_error_t doça_devinfo_get_ibdev_name (const doça_devinfo *devinfo, char *ibdev_name, uint32_t size)
Get the name of the IB device represented by a DOCA devinfo.

Parameters

**devinfo**
The device to query.

**ibdev_name**
The name of the IB device represented by devinfo.

**size**
The size of the input ibdev_name buffer, must be at least
DOCA_DEVINFO_IBDEV_NAME_SIZE which includes the null terminating byte.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

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

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

doca_error_t doca_devinfo_get_iface_name (const doca_devinfo *devinfo, char *iface_name, uint32_t size)
Get the name of the ethernet interface of a DOCA devinfo.

Parameters

- devinfo
  The device to query.
- iface_name
  The name of the ethernet interface of devinfo.
- size
  The size of the input iface_name buffer, must be at least DOCA_DEVINFO_IFACE_NAME_SIZE which includes the null terminating byte.

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

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

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

doca_error_t doca_devinfo_get_ipv4_addr (const doca_devinfo *devinfo, uint8_t *ipv4_addr, uint32_t size)
Get the IPv4 address of a DOCA devinfo.

Parameters

- devinfo
  The device to query.
**ipv4_addr**
The IPv4 address of devinfo.

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

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

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

**Description**
The IPv4 address type: uint8_t[DOCA_DEVINFO_IPV4_ADDR_SIZE].

doca_error_t doca_devinfo_get_ipv6_addr (const
doca_devinfo *devinfo, uint8_t *ipv6_addr, uint32_t size)
Get the IPv6 address of a DOCA devinfo.

**Parameters**

**devinfo**
The device to query.

**ipv6_addr**
The IPv6 address of devinfo.

**size**
The size of the input ipv6_addr buffer, must be at least DOCA_DEVINFO_IPV6_ADDR_SIZE

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

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

**Description**
The IPv6 address type: uint8_t[DOCA_DEVINFO_IPV6_ADDR_SIZE].
**docta_error_t**

**docta_devinfo_get_is_hotplug_manager_supported (const doca_devinfo *devinfo, uint8_t *is_hotplug_manager)**

Get the hotplug manager capability of a DOCA devinfo.

**Parameters**

*devinfo*

The device to query.

*is_hotplug_manager*

1 if the hotplug manager capability is supported, 0 otherwise.

**Returns**

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

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

**Description**

The hotplug manager property type: uint8_t*.

**docta_error_t**

**docta_devinfo_get_is_mmap_export_supported (const doca_devinfo *devinfo, uint8_t *mmap_export)**

Get the mmap export capability of the device.

**Parameters**

*devinfo*

The device to query.

*mmap_export*

**Returns**

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

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

**Description**

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

**doqa_devinfo_get_is_mmap_from_export_supported (const doqa_devinfo *devinfo, uint8_t *from_export)**

Get the mmap create from export capability of the device.

Parameters

- **devinfo**
  - The device to query.
- **from_export**
  - 1 if the mmap from export capability is supported, 0 otherwise.

Returns

- DOCA_SUCCESS - in case of success.
- Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_DRIVER - failed to query capability support.

Description

Get uint8_t value defining if the device can be used to create an mmap from an exported mmap. See `doqa_mmap_create_from_export()` in doqa_mmap.h true - device can be used with the mmap create from export API. false - create from export API is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.

**doqa_error_t doqa_devinfo_get_pci_addr (const doqa_devinfo *devinfo, doqa_pci_bdf *pci_addr)**

Get the PCI address of a DOCA devinfo.

Parameters

- **devinfo**
  - The device to query.
- **pci_addr**
  - The PCI address of devinfo.

Returns

- DOCA_SUCCESS - in case of success.
- Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_NOT_CONNECTED - provided devinfo does not support this property.
  - DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the PCI address from the OS.
**Description**

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

```c
#include <doctypes.h>
doca_error_t doca_devinfo_list_create (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_list_destroy()`

```c
#include <doctypes.h>
doca_error_t doca_devinfo_list_destroy (doca_devinfo **dev_list)
```

Destroy list of local device info structures.

**Parameters**

- `dev_list` : List to be destroyed.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

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

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

doca_error_t doca_devinfo_rep_get_is_list_all_supported(const doca_devinfo *devinfo, uint8_t *all_supported)
Get the representor devices discovery capability of the device.

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

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

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

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

Parameters

  devinfo
  The device to query.

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

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_DRIVER - failed to query capability support.

Description

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

doca_error_t

doca_devinfo_rep_get_is_list_nvme_supported (const
doca_devinfo *devinfo, uint8_t *nvme_supported)
Get the remote nvme discovery capability of the device.

Parameters

  devinfo
  The device to query.

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

Returns

DOCA_SUCCESS - in case of success. Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_DRIVER - failed to query capability support.
Description

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

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

Get the remote virtio blk discovery capability of the device.

Parameters

devinfo
The device to query.

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

Returns

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

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

Description

Get uint8_t value defining if the device can be used to create list of virtio blk remote devices. See doca_devinfo_remote_list_create(). true - device can be used with the remote list create API with filter DOCA_DEV_REMOTE_FILTER_VIRTIO_BLK. false - providing DOCA_DEV_REMOTE_FILTER_VIRTIO_BLK is guaranteed to fail with DOCA_ERROR_NOT_SUPPORTED.
doса_error_t
doca_devinfo_rep_get_is_list_virtio_fs_supported (const
doca_devinfo *devinfo, uint8_t *virtio_fs_supported)
Get the remote virtio fs discovery capability of the device.

Parameters

**devinfo**
The device to query.

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

Returns

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

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

Description

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

doса_error_t
doca_devinfo_rep_get_is_list_virtio_net_supported (const
doca_devinfo *devinfo, uint8_t *virtio_net_supported)
Get the remote virtio net discovery capability of the device.

Parameters

**devinfo**
The device to query.

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

Returns

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

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

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

doca_error_t doca_devinfo_rep_get_pci_addr (const doca_devinfo_rep *devinfo_rep, doca_pci_bdf *pci_addr)
Get the PCI address of a DOCA devinfo_rep.

Parameters
- devinfo_rep
  The representor of device to query.
- pci_addr
  The PCI address of the devinfo_rep.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
The PCI address type: struct doca_pci_bdf.

doca_error_t doca_devinfo_rep_get_pci_func_type (const doca_devinfo_rep *devinfo_rep, doca_pci_func_type *pci_func_type)
Get the PCI function type of a DOCA devinfo_rep.

Parameters
- devinfo_rep
  The representor of device to query.
- pci_func_type
  The PCI function type of the devinfo_rep.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
The pci function type: enum doca_pci_func_type.

doca_error_t doca_devinfo_rep_get_vuid (const
doca_devinfo_rep *devinfo_rep, char *rep_vuid, uint32_t size)
Get the Vendor Unique ID of a representor DOCA devinfo.

Parameters

- devinfo_rep
  The representor device to query.

- rep_vuid
  The Vendor Unique ID of devinfo_rep.

- size
  The size of the vuid buffer, including the terminating null byte (\"\").

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

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

doca_error_t doca_devinfo_rep_list_create (doca_dev *dev, int filter, doca_devinfo_repdev_list_rep, uint32_t *nb_devs_rep)
Create list of available representor devices accessible by dev.

Parameters

dev
  Local device with access to representors.
filter
   Bitmap filter of representor types. See enum doca_dev_rep_filter for more details.

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

nb_devs_rep
   Number of available representor devices.

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory for list.
- DOCA_ERROR_NOT_SUPPORTED - local device does not expose representor devices.

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

Note:
Returned list must be destroyed using doca_devinfo_rep_list_destroy()

doca_error_t doca_devinfo_rep_list_destroy(doca_devinfo_rep **dev_list_rep)
Destroy list of representor device info structures.

Parameters
dev_list_rep
   List to be destroyed.

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

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

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

Core
DOCA API for integration with DPDK.

doca_error_t doca_dpdk_port_as_dev (uint16_t port_id, doca_dev **dev)

Return the DOCA device associated with a DPDK port.

Parameters

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

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

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

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

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

Attach a DPDK port specified by DOCA device.

Parameters

dev
DOCA device to attach PDK port for.

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

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

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_DRIVER - in case of DPDK error during DPDK port attach.
- DOCA_ERROR_NO_MEMORY - in case of memory allocation failure. TODO: complete error documentation
Description
Thread unsafe API.
It’s the user responsibility to set the DPDK EAL initialization to skip probing the PCI device
associated with the given DOCA device to prevent EAL from using it.
No initialization is done for the probed PDPK port and the port is not started.

2.4.6. DOCA Error

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

```c
const __DOCA_EXPERIMENTAL char
*doca_get_error_name (doca_error_t error)
```

Returns the string representation of an error code name.

Parameters

- `error` - Error code to convert to string.

Returns

char* pointer to a NULL-terminated string.

Description

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

```c
const __DOCA_EXPERIMENTAL char
*doca_get_error_string (doca_error_t error)
```

Returns the description string of an error code.

Parameters

- `error` - Error code to convert to description string.

Returns

char* pointer to a NULL-terminated string.
Description
This function returns the description string of an error code. If the error code is not recognized, “unrecognized error code” is returned.

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

Save the first encountered `doca_error_t`.

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

2.4.7. DOCA Hotplug
Core
DOCA API for hot plug/un-plug devices.

`doca_error_t doca_dev_rep_hotplug (const doca_dev_hotplug_attr *attr, doca_dev_rep **dev_rep)`
Hotplug and initialize representor device for use.

Parameters
- `attr`
  DOCA hotplug attr with designated characteristics.

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

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

`doca_error_t doca_dev_rep_hotunplug (doca_dev_rep *rep_dev)`
Destroy and unplug representor device instance.

Parameters
- `rep_dev`
  The previously hotplugged representor doca device instance.
Returns

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

Description

Note:

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

2.4.8. DOCA Memory Map

Core

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

typedef void (doca_mmap_memrange_free_cb_t)

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

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

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

Parameters

user_data

mmap

   DOCA memory map structure with default/unset attributes.

Returns

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

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

Description

The returned memory map object can be manipulated with doca_mmap_property_set[] API.
Once all required mmap attributes set it should be reconfigured and adjusted to meet object size setting with \texttt{doxa\_mmap\_start}. See \texttt{doxa\_mmap\_start} for the rest of the details.

\texttt{doxa\_error\_t doxa\_mmap\_create\_from\_export (const doca\_data *user\_data, const void *export\_desc, size\_t export\_desc\_len, doca\_dev *dev, doca\_mmap **mmap)}

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

\textbf{Parameters}

\begin{itemize}
  \item \texttt{user\_data}
  \item \texttt{export\_desc} An export descriptor generated by \texttt{doxa\_mmap\_export}.
  \item \texttt{export\_desc\_len} Length in bytes of the \texttt{export\_desc}.
  \item \texttt{dev} A local device connected to the device that resides in the exported mmap. Device must have from export capability. See \texttt{doxa\_dev\_info\_get\_is\_mmap\_from\_export\_supported} in \texttt{doxa\_dev.h}
  \item \texttt{mmap} DOCA memory map granting access to remote memory.
\end{itemize}

\textbf{Returns}

DOCA\_SUCCESS - in case of success. doca\_error code - in case of failure:

\begin{itemize}
  \item DOCA\_ERROR\_INVALID\_VALUE - if an invalid input had been received or internal error. The following errors are internal and will occur if failed to produce new mmap from export descriptor:
    \item DOCA\_ERROR\_NO\_MEMORY - if internal memory allocation failed.
    \item DOCA\_ERROR\_NOT\_SUPPORTED - device missing create from export capability.
    \item DOCA\_ERROR\_NOT\_PERMITTED
    \item DOCA\_ERROR\_DRIVER
\end{itemize}

\textbf{Description}

Once this function called on the object it considered as from\_export.

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

\begin{itemize}
  \item Setting the properties of the mmap using \texttt{doxa\_mmap\_property\_set}.
  \item Adding a device to the mmap using \texttt{doxa\_mmap\_dev\_add}.
  \item Removing a device to the mmap using \texttt{doxa\_mmap\_dev\_rm}.
  \item Adding a memory range to the mmap using \texttt{doxa\_mmap\_populate}.
\end{itemize}
Exporting the mmap using `doa_mmap_export()`.

**Note:**

The created object not backed by local memory.

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

`doa_error_t doa_mmap_destroy (doa_mmap *mmap)`

Destroy DOCA Memory Map structure.

**Parameters**

- **mmap**
  
The DOCA memory map structure.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doa_error code** - in case of failure:
  
  - **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received.
  
  - **DOCA_ERROR_NOT_PERMITTED** - if there is a memory region pointed by one or more `struct doca_buf`, or if memory deregistration failed.

**Description**

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

`doa_error_t doa_mmap_dev_add (doa_mmap *mmap, doa_dev *dev)`

Register DOCA memory map on a given device.

**Parameters**

- **mmap**
  
  DOCA memory map structure.

- **dev**
  
  DOCA Dev instance with appropriate capability.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doa_error code** - in case of failure:
  
  - **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received.
  
  - **DOCA_ERROR_NOT_PERMITTED** - if memory deregistration failed or the operation is not premitted for the given mmap (see details in this function description).
- DOCA_ERROR_NO_MEMORY - if reached to DOCA_MMAP_MAX_NUM_DEVICES.
- DOCA_ERROR_IN_USE - if doca_dev already exists in doca_mmap.

**Description**

This operation is not permitted for:

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

```c
#include <doacore.h>

doca_error_t doca_mmap_dev_rm (doca_mmap *mmap, doca_dev *dev)
```

Deregister given device from DOCA memory map.

**Parameters**

- **mmap**
  
  DOCA memory map structure.

- **dev**
  
  DOCA Dev instance that was previously added.

**Returns**

- DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  
  - DOCA_ERROR_INVALID_VALUE - if an invalid input had been received or doca_dev doesn’t exists in doca_mmap.
  - DOCA_ERROR_NOT_PERMITTED - if memory deregistration failed or the operation is not premitted for the given mmap (see details in this function description).

**Description**

This operation is not permitted for:

- un-started/stopped memory map object.
- memory map object that have been exported or created from export.
The function doca_error_t doca_mmap_export (doca_mmap *mmap, const doca_dev *dev, void **export_desc, size_t *export_desc_len)
Compose memory map representation for later import with doca_mmap_create_from_export() for one of the devices previously added to the memory map.

Parameters

mmap
DOCA memory map structure.

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

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

export_desc_len
Length in bytes of the export_desc.

Returns

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

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

Description

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

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

This operation is not permitted for:

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

The following are NOT possible after export:

- Setting the properties of the mmap using doca_mmap_property_set().
- Adding a device to the mmap using `doxa_mmap_dev_add()`.
- Removing a device from the mmap using `doxa_mmap_dev_rm()`.
- Adding a memory range to the mmap using `doxa_mmap_populate()`.
- Exporting the mmap using `doxa_mmap_export()`.

```c
#include <doxa.h>

doxa_error_t doxa_mmap_get_exported (doxa_mmap *mmap, uint8_t *exported)
```

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

**Parameters**

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

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doxa_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received.

```c
#include <doxa.h>

doxa_error_t doxa_mmap_get_from_export (doxa_mmap *mmap, uint8_t *from_export)
```

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

**Parameters**

- **mmap**
  - The DOCA memory map structure.
- **from_export**
  - 1 if mmap had been created from export, 0 otherwise.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doxa_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - if an invalid input had been received.


**doca_error_t doca_mmap_get_max_num_chunks (doca_mmap *mmap, uint32_t *max_num_chunks)**

Get the max number of chunks to populate in a DOCA Memory Map.

**Parameters**

- **mmap**
  
  The DOCA memory map structure.

- **max_num_chunks**
  
  The max number of chunks to populate in mmap.

**Returns**

- DOCA_SUCCESS - in case of success.
- doca_error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

**doca_error_t doca_mmap_get_max_num_devices (doca_mmap *mmap, uint32_t *max_num_devices)**

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

**Parameters**

- **mmap**
  
  The DOCA memory map structure.

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

**Returns**

- DOCA_SUCCESS - in case of success.
- doca_error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

**doca_error_t doca_mmap_get_numbufs (doca_mmap *mmap, uint32_t *num_bufs)**

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

**Parameters**

- **mmap**
  
  The DOCA memory map structure.

- **num_bufs**
  
  The total number of `struct doca_buf` objects pointing to the memory in mmap.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

doca_error_t doca_mmap_get_user_data (doca_mmap *mmap, doca_data *user_data)
Get the user_data of a DOCA Memory Map.

Parameters
mmap
The DOCA memory map structure.
user_data
The user_data of mmap.

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.

Description

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

doca_error_t doca_mmap_populate (doca_mmap *mmap, void *addr, size_t len, size_t pg_sz,
doca_mmap_memrange_free_cb_t *free_cb, void *opaque)
Add memory range to DOCA memory map.

Parameters
mmap
DOCA memory map structure.
addr
Start address of the memory range to be populated.
len
The size of the memory range in bytes.
pg_sz
Page size alignment of the provided memory range. Must be >= 4096 and a power of 2.
**free_cb**
Callback function to free the populated memory range on memory map destroy.

**opaque**
Opaque value to be passed to free_cb once called.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if doca_mmap status is invalid for this operation or device registration failed or addr and len intersect with an existing chunk.
- DOCA_ERROR_NO_MEMORY - if reached to DOCA_MMAP_MAX_NUM CHUNKS, or memory allocation failed.

**Description**

This operation is not permitted for:

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

**doca_error_t doca_mmap_set_max_num_chunks (doca_mmap *mmap, uint32_t max_num_chunks)**

Set a new max number of chunks to populate in a DOCA Memory Map. Note: once a memory map object has been first started this functionality will not be available.

**Parameters**

**mmap**
The DOCA memory map structure.

**max_num_chunks**
The new max number of chunks to populate in mmap.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - if an invalid input had been received.
- DOCA_ERROR_NOT_PERMITTED - if trying to set the max number of chunks after first start of the mmap.
**do**ca_error_t** do**ca_mmap_set_max_num_devices
(**do**ca_mmap *mmap, uint32_t max_num_devices)
Set a new max number of devices to add to a DOCA Memory Map. Note: once a memory map object has been first started this functionality will not be available.

**Parameters**

**mmap**
The DOCA memory map structure.

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

**Returns**

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

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

**do**ca_error_t** do**ca_mmap_start (**do**ca_mmap *mmap)
Start DOCA Memory Map.

**Parameters**

**mmap**
DOCA memory map structure.

**Returns**

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

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

**Description**

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

The following become possible only after start:

- Adding a device to the mmap using **do**ca_mmap_dev_add().
- Removing a device to the mmap using **do**ca_mmap_dev_rm().
- Adding a memory range to the mmap using **do**ca_mmap_populate().
- Exporting the mmap using **do**ca_mmap_export().
- Mapping doca_buf structures to the memory ranges in the using `doca_buf_inventory_buf_by_addr()` or `doca_buf_inventory_buf_dup()`.

The following are NOT possible after the first time `start` is called:
- Setting the properties of the mmap using `doca_mmap_property_set()`.

```c
doca_error_t doca_mmap_stop (doca_mmap *mmap)
```

Stop DOCA Memory Map.

**Parameters**
- `mmap`
  - DOCA memory map structure.

**Returns**
- `DOCA_SUCCESS` - in case of success.
- `doca_error` code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - if an invalid input had been received.

**Description**
Prevents execution of different operations on the mmap. For details see `doca_mmap_start()`.

### 2.4.9. DOCA RDMA Core

DOCA RDMA bridge.

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

Get the protection domain associated with a DOCA device.

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

**Returns**
- `DOCA_SUCCESS` - in case of success.
- `doca_error` code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - in case of invalid input.
  - `DOCA_ERROR_BAD_STATE` - in case the device’s pd is not valid (bad state)
2.4.10. DOCA Types

Core
DOCA Types introduces types that are common for many libraries.

```c
struct doca_pci_bdf
```

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

```c
enum doca_access_flags
```

Specifies the permission level for DOCA buffer.

**Values**

- `DOCA_ACCESS_LOCAL_READ = 0`
- `DOCA_ACCESS_LOCAL_WRITE = 1`
- `DOCA_ACCESS_REMOTE_WRITE = (1<<1)`
- `DOCA_ACCESS_REMOTE_READ = (1<<2)`
- `DOCA_ACCESS_REMOTE_ATOMIC = (1<<3)`

```c
enum doca_pci_func_type
```

Specifies the PCI function type for DOCA representor device.

**Values**

- `DOCA_PCI_FUNC_PF = 0`
- `DOCA_PCI_FUNC_VF`
- `DOCA_PCI_FUNC_SF`

2.5. Comm Channel

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

```c
enum doca_comm_channel_msg_flags
```

Flags for send/receive functions.

**Values**

- `DOCA_CC_MSG_FLAG_NONE = 0`
typedef HANDLE doca_event_channel_t
endpoint notification file descriptor for blocking with epoll() for recv ready event
< Windows

doca_error_t doca_comm_channel_ep_connect
(doca_comm_channel_ep_t *local_ep, const char *name, doca_comm_channel_addr_t **peer_addr)
Client side Connect.

Parameters
local_ep
handle for the endpoint created beforehand with doca_comm_channel_ep_create().
name
identifies the service. Use doca_comm_channel_get_max_service_name_len() to get the maximal service name length.
peer_addr
handle to use for sending packets and recognize source of messages.

Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if no ep object, name or peer_address pointer given. DOCA_ERROR_NOT_PERMITTED if the function was called on the service or 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). DOCA_ERROR_DRIVER if acquiring device attributes 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.
`doxa_error_t doca_comm_channel_ep_create (doxa_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.

`doxa_error_t doca_comm_channel_ep_destroy (doxa_comm_channel_ep_t *local_ep)`

Release endpoint handle.

**Parameters**

- **local_ep**
  - handle for the endpoint created beforehand with `doxa_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 `doxa_comm_channel_ep_disconnect()` is included as part of the destroy process.
**doca_error_t doca_comm_channel_ep_disconnect**

*doca_comm_channel_ep_t *local_ep,*

*doca_comm_channel_addr_t *peer_addr)*

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

**Parameters**

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

- **peer_addr**
  peer address to be disconnect from.

**Returns**

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if no ep was provided.

DOCA_ERROR_NOT_CONNECTED if there is no connection.

**doca_error_t**

**doca_comm_channel_ep_event_handle_arm_recv**

*doca_comm_channel_ep_t *local_ep)*

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

**Parameters**

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

**Returns**

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if no ep object was given.
**doqa_error_t**

**doqa_comm_channel_ep_event_handle_arm_send**

(docaq_comm_channel_ep_t *local_epi)

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_epi

handle for the endpoint created beforehand with **doqa_comm_channel_ep_create()**.

**Returns**

DOCA_SUCCESS on success DOCA_ERROR_INVALID_VALUE if no ep object was given.

**doqa_error_t** **doqa_comm_channel_ep_get_device**

(docaq_comm_channel_ep_t *ep, doqa_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.

**doqa_error_t**

**doqa_comm_channel_ep_get_device_rep**

(docaq_comm_channel_ep_t *ep, doqa_dev_rep **device_rep)

get device representor property of endpoint.

**Parameters**

ep

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

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

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

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

Parameters
local_ep
    handle for the endpoint created beforehand with doca_comm_channel_ep_create().
send_event_channel
    handle for send event channel.
recv_event_channel
    handle for receive event channel.

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

doca_error_t
doca_comm_channel_ep_get_max_msg_size
doca_comm_channel_ep_t *ep, uint16_t *max_msg_size

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

Parameters

ep
  endpoint from which the property should be retrieved.
max_msg_size
  maximal msg size used by the endpoint.

Returns

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

doca_error_t
doaca_comm_channel_ep_get_recv_queue_size
(doca_comm_channel_ep_t *ep, uint16_t
*recv_queue_size)
get receive queue size property of endpoint. The size returned is the actual size being used and might differ from the size set with doaca_comm_channel_ep_set_recv_queue_size(), as there is a minimal size requirement and the size is rounded up to the closest power of 2. If receive queue size was not set, using doaca_comm_channel_ep_set_recv_queue_size(), a default value is used and can be inquired by calling doaca_comm_channel_ep_get_recv_queue_size().

Parameters

ep
  endpoint from which the property should be retrieved.
recv_queue_size
  receive queue size used by the endpoint.

Returns

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

doca_error_t
doaca_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 doaca_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.

```c
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_BAD_STATE if no doca_dev or no doca_dev_rep was set.
- DOCA_ERROR_NOT_PERMITTED if the function was called on the client side or 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.
docta_error_t docta_comm_channel_ep_recvfrom(doca_comm_channel_ep_t *local_ep, void *msg, size_t *len, int flags, docta_comm_channel_addr_t **peer_addr)

Receive message from connected client/service.

Parameters

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

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

len
flags
    flag for receive command. currently no flags are supported.

peer_addr
    received message source address handle

Returns

DOCA_SUCCESS on successful receive. If a message was received, the value pointed by len will be updated with the number of bytes received. DOCA_ERROR_INVALID_VALUE if any of the parameters is NULL. DOCA_ERROR_NOT_CONNECTED if endpoint is service and listen was not called. DOCA_ERROR_AGAIN if no message was received. when returned, the user can use the endpoint’s docta_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, docta_comm_channel_ep_recvfrom also used for accepting new connection from clients.
doxa_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.

doса_error_t doса_comm_channel_ep_set_device
(doca_comm_channel_ep_t *ep, doса_dev *device)
set device property for endpoint.

Parameters
ep
endpoint to set the property for.
device
device to use in endpoint.
Returns
DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

`doca_error_t
doca_comm_channel_ep_set_device_rep
doca_comm_channel_ep_t *ep, doca_dev_rep *device_rep)
set device representor property for endpoint.

Parameters

ep
document to set the property for.

device_rep
device representor to use in endpoint.

Returns
DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

`doca_error_t
doca_comm_channel_ep_set_max_msg_size
doca_comm_channel_ep_t *ep, uint16_t max_msg_size)
set maximal msg size property for endpoint. The value max_msg_size may be increased internally, the actual value can be queried using doca_comm_channel_ep_get_max_msg_size().

Parameters

ep
document to set the property for.

max_msg_size
maximal msg size to use in endpoint.

Returns
DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.
doca_error_t
doca_comm_channel_ep_set_recv_queue_size
(doca_comm_channel_ep_t *ep, uint16_t
recv_queue_size)
set receive queue size property for endpoint. The value recv_queue_size
may be increased internally, the actual value can be queried using
doca_comm_channel_ep_get_recv_queue_size().

Parameters
ep
    endpoint to set the property for.
recv_queue_size
    receive queue size to use in endpoint.

Returns
DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid
parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.

doca_error_t
doca_comm_channel_ep_set_send_queue_size
(doca_comm_channel_ep_t *ep, uint16_t
send_queue_size)
set send queue size property for endpoint. The value send_queue_size
may be increased internally, the actual value can be queried using
doca_comm_channel_ep_get_send_queue_size().

Parameters
ep
    endpoint to set the property for.
send_queue_size
    send queue size to use in endpoint.

Returns
DOCA_SUCCESS if property set successfully. DOCA_ERROR_INVALID_VALUE if an invalid
parameter was given. DOCA_ERROR_BAD_STATE if endpoint is already active.
**docta_error_t**

**docta_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.

**docta_error_t**

**docta_comm_channel_get_max_recv_queue_size**
(doca_devinfo *devinfo, uint32_t *max_recv_queue_size)

Get the maximum receive queue size supported by comm_channel.

**Parameters**

- **devinfo**
  - devinfo that should be inquired for its maximum receive queue size under comm channel limitations.
- **max_recv_queue_size**
  - the maximum receive queue size supported by comm_channel.

**Returns**

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or max_recv_queue_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.
doxa_error_t
doca_comm_channel_get_max_send_queue_size
(doca_devinfo *devinfo, uint32_t *
max_send_queue_size)
Get the maximum send queue size supported by comm_channel.

Parameters
devinfo
devinfo that should be inquired for its maximum send queue size under comm_channel
limitations.
max_send_queue_size
the maximum send queue size supported by comm_channel.

Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or
max_send_queue_size is NULL. DOCA_ERROR_UNEXPECTED if an unexpected error
occurred.

doca_error_t
doca_comm_channel_get_max_service_name_len
(uint32_t *max_service_name_len)
Get the comm_channel maximum Service name length.

Parameters
max_service_name_len
The comm_channel max service name length, including the terminating null byte [""].

Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if max_service_name_len is
NULL.
doca_error_t
doca_comm_channel_get_service_max_num_connections
doca_devinfo *devinfo, uint32_t *
max_num_connections)
Get the maximum number of connections the service can hold.

Parameters

devinfo
  devinfo that should be inquired for its maximum number of connections.
max_num_connections
  the maximum number of connections the service can hold.

Returns

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if either devinfo or
max_num_connections is NULL. DOCA_ERROR_NOT_SUPPORTED if querying this capability
is not supported by the device. DOCA_ERROR_UNEXPECTED if an unexpected error occurred.

Description

Note:
This capability should be queried only on the service side.

Note:

This capability should be queried only on the service side.

doca_error_t
doca_comm_channel_peer_addr_get_recv_bytes
(const doca_comm_channel_addr_t *peer_addr,
uint64_t *recv_bytes)
get total bytes received from specific peer address

Parameters

peer_addr
  Pointer to peer_addr to query statistics for.
recv_bytes
  Will contain the number of received bytes from the given peer_addr.
Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description
This function will return the total number of bytes received from a given peer_addr, updated to the last time `doca_comm_channel_peer_addr_update_info()` was called.

doca_error_t
doca_comm_channel_peer_addr_get_recv_messages
(const doca_comm_channel_addr_t *peer_addr,
uint64_t *recv_messages)
get total messages received from specific peer address

Parameters
peer_addr
  Pointer to peer_addr to query statistics for.
recv_messages
  Will contain the number of received messages from the given peer_addr.

Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if any of the arguments are NULL.

Description
This function will return the total number of messages received from a given peer_addr, updated to the last time `doca_comm_channel_peer_addr_update_info()` was called.
```c

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.

```
**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.

```c
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.

```c
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.

```c

doca_error_t
doca_comm_channel_peer_addr_set_user_data
(doca_comm_channel_addr_t *peer_addr, uint64_t
user_context)
```

Save 'user_context' in peer_addr handle.

**Parameters**

- **peer_addr**
  
  Pointer to peer_addr to set user_context to.

- **user_context**

  Data to set for peer_addr.

**Returns**

DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_address is NULL.

**Description**

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

```c

doca_error_t
doca_comm_channel_peer_addr_update_info
(doca_comm_channel_addr_t *peer_addr)
```

update statistics for given peer_addr

**Parameters**

- **peer_addr**

  Pointer to peer_addr to update statistics in.
Returns
DOCA_SUCCESS on success. DOCA_ERROR_INVALID_VALUE if peer_addr is NULL.
DOCA_ERROR_CONNECTION_INPROGRESS if connection is not yet established.
DOCA_ERROR_CONNECTION_ABORTED if the connection failed.

Description
Should be used before calling to any peer_addr information function to update the saved
statistics. This function can also be used to check if connection to a given peer_addr
is currently connected. If a connection has failed, it is the user’s responsibility to call
doca_comm_channel_ep_disconnect() to free the peer_addr resources.

2.6. Compatibility Management

Lib to define compatibility with current version, define experimental Symbols.
To set a Symbol (or specifically a function) as experimental:
__DOCA_EXPERIMENTAL int func_declare(int param1, int param2);
To remove warnings of experimental compile with "-D DOCA_ALLOW_EXPERIMENTAL_API"

#define __DOCA_EXPERIMENTAL
__declspec(deprecated("Symbol is defined as
experimental"), DLL_EXPORT_ATTR)
To set a Symbol (or specifically a function) as experimental.

#define DOCA_STRUCT_START uint32_t
__doca_api_version

Compatibility Helpers

2.7. DOCA COMPRESS engine

DOCA COMPRESS library. For more details please refer to the user guide on DOCA devzone.
struct doca_compress_job

enum doca_compress_job_types

Available jobs for DOCA COMPRESS.

Values

DOCA_COMPRESS_DEFLATE_JOB = DOCA_ACTION_COMPRESS_FIRST+1
DOCA_DECOMPRESS_DEFLATE_JOB

__DOCA_EXPERIMENTAL doca_ctx

*doca_compress_as_ctx (doca_compress *compress)

Parameters

compress

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

Returns

Non NULL upon success, NULL otherwise.

Description

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

doca_error_t doca_compress_create (doca_compress **compress)

Parameters

compress

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - compress argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_compress.
- DOCA_ERROR_INITIALIZATION - failed to initialize a mutex.
Description
Create a DOCA COMPRESS instance.

doca_error_t doca_compress_destroy
doca_compress_destroy
(doca_compress *compress)

Parameters
compress
    Pointer to instance to be destroyed.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
    ➤ DOCA_ERROR_IN_USE - if unable to gain exclusive access to the compress instance or if
        one or more work queues are still attached. These must be detached first.

Description
Destroy a DOCA COMPRESS instance.

doca_error_t doca_compress_get_max_buffer_size
(doca_compress_get_max_buffer_size
(doca_compression_devinfo *devinfo,
doca_compress_job_types job_type, uint32_t
max_buffer_size)

Parameters
devinfo
    The DOCA device information
job_type
    doca_compress job type. See enum doca_compress_job_types.
max_buffer_size
    The max buffer size for DOCA COMPRESS operation in bytes.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
    ➤ DOCA_ERROR_INVALID_VALUE - received invalid input.
- **DOCA_ERROR_NOT_SUPPORTED** - failed to query device capabilities. or provided devinfo does not support the given doca_compress job.

**Description**
Get maximum buffer size for DOCA COMPRESS job.

```c
typedef enum
doca_error_t

doca_compress_get_max_list_buf_num_elem (const
doctr_devinfo *devinfo, uint32_t *max_list_num_elem)
```

**Parameters**
- **devinfo**
  - The DOCA device information.
- **max_list_num_elem**
  - The maximum supported number of elements in DOCA linked-list buffer. The value 1 indicates that only a single element is supported.

**Returns**
- DOCA_SUCCESS - in case of success.
- doca_error code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - in case of invalid input.

**Description**
Get the maximum supported number of elements in DOCA linked-list buffer for compress job.

```c
typedef enum
doca_error_t
doca_compress_job_get_supported
doctr_devinfo *devinfo, doctr_compress_job_types
job_type)
```

**Parameters**
- **devinfo**
  - The DOCA device information
- **job_type**
  - doca_compress job type. See enum doca_compress_job_types.

**Returns**
- DOCA_SUCCESS - in case the job is supported.
- doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_compress job.

**Description**

Check if given device is capable for given doca_compress job.

### 2.8. Environment Configurations

```c
#define DOCA_COMPAT_HELPERS
defines the support/need for compatibility helper utils
```

### 2.9. ct

DOCA HW connection tracking library.

DOCA HW offload flow library. For more details please refer to the user guide on DOCA devzone.
struct doca_ct_cfg
doca ct global configuration

struct doca_flow_action_desc
action description

struct doca_flow_action_descs
action descriptions

struct doca_flow_action_descs_meta
Metadata action description per field.

struct doca_flow_action_field
extended modification action

struct doca_flow_actions
doca flow actions information

struct doca_flow_aged_query
aged flow query callback context

struct doca_flow_cfg
doca flow global configuration

struct doca_flow_encap_action
doca flow encap data information

struct doca_flow_error
doca flow error message struct

struct doca_flow_fwd
forwarding configuration

struct doca_flow_match
doca flow matcher information
struct doca_flow_meta
  doca flow meta data

struct doca_flow_monitor
  doca monitor action configuration

struct doca_flow_ordered_list

struct doca_flow_pipe_attr
  pipe attributes

struct doca_flow_pipe_cfg
  pipeline configuration

struct doca_flow_port_cfg
  doca flow port configuration

struct doca_flow_query
  flow query result

struct doca_flow_resource_crypto_cfg
  doca flow crypro resource configuration

struct doca_flow_resource_meter_cfg
  doca flow meter resource configuration

struct doca_flow_resource_rss_cfg
  doca flow rss resource configuration

struct doca_flow_resources
  doca flow resource quota

struct doca_flow_shared_resource_cfg
  doca flow shared resource configuration

struct doca_flow_shared_resource_result
  flow shared resources query result
enum doca_ct_flags
CT flags.

Values
- **DOCA_CT_FLAG_STATS** = 1u<<0
  Enable counter for internal pipes
- **DOCA_CT_FLAG_BYPASS** = 1u<<1
  Bypass internal 6-tuples pipe with regular pipe

enum doca_ct_types
CT flags.

Values
- **DOCA_CT_TYPE_KNOWN**
  Known TCP/UDP connection
- **DOCA_CT_TYPE_SYN**
  New connection packet
- **DOCA_CT_TYPE_FIN**
  Connection finish packet
- **DOCA_CT_TYPE_RST**
  Connection reset packet

enum doca_flow_action_type
action type enumeration

Values
- **DOCA_FLOW_ACTION_AUTO** = 0
- **DOCA_FLOW_ACTION_CONSTANT**
- **DOCA_FLOW_ACTION_SET**
- **DOCA_FLOW_ACTION_ADD**
- **DOCA_FLOW_ACTION_COPY**
- **DOCA_FLOW_ACTION_MAX**

enum doca_flow_entry_op
doca flow entry operation

Values
- **DOCA_FLOW_ENTRY_OP_ADD**
  Add entry
- **DOCA_FLOW_ENTRY_OP_DEL**
enum doca_flow_entry_status
doca flow entry status

Values
DOCA_FLOW_ENTRY_STATUS_IN_PROCESS
DOCA_FLOW_ENTRY_STATUS_SUCCESS
DOCA_FLOW_ENTRY_STATUS_ERROR

enum doca_flow_error_type
doca flow error type define

Values
DOCA_FLOW_ERROR_UNKNOWN
Unknown error
DOCA_FLOW_ERROR_UNSUPPORTED
Operation unsupported
DOCA_FLOW_ERROR_INVALID_PARAM
Invalid parameter
DOCA_FLOW_ERROR_PIPE_BUILD_ITEM
Build pipe match items error
DOCA_FLOW_ERROR_PIPE_MODIFY_ITEM
Modify pipe match items error
DOCA_FLOW_ERROR_PIPE_BUILD_ACTION
Build pipe actions error
DOCA_FLOW_ERROR_PIPE_MODIFY_ACTION
Modify pipe actions error
DOCA_FLOW_ERROR_PIPE_BUILD_FWD
Build pipe fwd error
DOCA_FLOW_ERROR_FLOW_CREATE
Flow creation error
DOCA_FLOW_ERROR_FLOW_DESTROY
Flow destroy error
DOCA_FLOW_ERROR_OOM
Out of memory
DOCA_FLOW_ERROR_PORT
Port error
DOCA_FLOW_ERROR_VERIFY_CONFIG
Verification error
enum doca_flow_flags_type
doca flow flags type

Values

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

enum doca_flow_fwd_type
forwarding action type

Values

DOCA_FLOW_FWD_NONE = 0
    No forward action be set
DOCA_FLOW_FWD_RSS
    Forwards packets to rss
DOCA_FLOW_FWD_PORT
    Forwards packets to one port
DOCA_FLOW_FWD_PIPE
    Forwards packets to another pipe
DOCA_FLOW_FWD_DROP
    Drops packets
DOCA_FLOW_FWD_ORDERED_LIST_PIPE
    Forwards packet to a specific entry in an ordered list pipe.

enum doca_flow_match_tcp_flags
doca flow match flags

Values

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

DOCA_FLOW_MATCH_TCP_FLAG_ECE = (1<<6)
match tcp packet with Urg flag

DOCA_FLOW_MATCH_TCP_FLAG_CWR = (1<<7)
match tcp packet with Urg flag

enum doca_flow_ordered_list_element_type

Type of an ordered list element.

Values

DOCA_FLOW_ORDERED_LIST_ELEMENT_ACTIONS
    Ordered list element is struct doca_flow_actions, the next element is struct
doca_flow_action_descs associated with the current element.

DOCA_FLOW_ORDERED_LIST_ELEMENT_ACTION_DESCS
    Ordered list element is struct doca_flow_action_descs. If the previous element type is
    ACTIONS, the current element is associated with it. Otherwise the current element is
    ordered w.r.t. the previous one.

DOCA_FLOW_ORDERED_LIST_ELEMENT_MONITOR
    Ordered list element is struct doca_flow_monitor.

enum doca_flow_pipe_type

doca flow pipe type

Values

DOCA_FLOW_PIPE_BASIC
    Flow pipe

DOCA_FLOW_PIPE_CONTROL
    Control pipe

DOCA_FLOW_PIPE_LPM
    longest prefix match (LPM) pipe

DOCA_FLOW_PIPE_ORDERED_LIST
    Ordered list pipe

enum doca_flow_port_type

doca flow port type

Values

DOCA_FLOW_PORT_DPDK_BY_ID
    dpdk port by mapping id
enum doca_flow_shared_resource_type
Shared resource supported types.

Values

DOCA_FLOW_SHARED_RESOURCE_METER
Shared meter type
DOCA_FLOW_SHARED_RESOURCE_COUNT
Shared counter type
DOCA_FLOW_SHARED_RESOURCE_RSS
Shared rss type
DOCA_FLOW_SHARED_RESOURCE_CRYPTO
Shared crypto action type
DOCA_FLOW_SHARED_RESOURCE_MAX
Shared max supported types

enum doca_rss_type
rss offload types

Values

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

typedef (*doca_flow_entry_process_cb)
   (doca_flow_pipe_entry* entry, enum
doca_flow_entry_status status, enum
doca_flow_entry_op op, void* user_ctx)
doca flow entry process callback

typedef (*doca_flow_shared_resource_unbind_cb)
   (enum doca_flow_shared_resource_type, uint32_t
shared_resource_id, void* bindable_obj)
doca flow shared resource unbind callback
DOCA Libraries API

__DOCA_EXPERIMENTAL void doca_ct_destroy (void)
Destroy the doca ct.

Description
Release all the resources used by doca ct.
Must be invoked before doca flow destroy.

__DOCA_EXPERIMENTAL int doca_ct_init (const
doca_ct_cfg *cfg)
Initialize the doca ct.

Parameters
cfg
   CT configuration.

Returns
0 on success, a negative errno value otherwise.

Description
This is the global initialization function for doca ct. It initializes all resources used by doca
flow.
Must be invoked first before any other function in this API. this is a one time call, used for doca
ct initialization and global configurations.
Must be invoked after Doca Flow initilization, before port start.

__DOCA_EXPERIMENTAL int doca_flow_aging_handle
(doca_flow_port *port, uint16_t queue, uint64_t quota,
doca_flow_aged_query *entries, int len)
Handle aging of flows in queue.

Parameters
port
   Port to handle aging
queue
   Queue identifier.
**quota**
Max time quota in micro seconds for this function to handle aging.

**entries**
User input entries array for the aged flows.

**len**
User input length of entries array.

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

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

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

```c
__DOCA_EXPERIMENTAL void doca_flow_destroy (void)
```
Destroy the doca flow.

**Description**
Release all the resources used by doca flow.
Must be invoked at the end of the application, before it exits.

```c
__DOCA_EXPERIMENTAL int
doca_flow_entries_process (doca_flow_port *port, uint16_t pipe_queue, uint64_t timeout, uint32_t max_processed_entries)
```
Process entries in queue.

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

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

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

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

Initialize the doca flow.

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

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

Description
   This is the global initialization function for doca flow. It initializes all resources used by doca
   flow.
   Must be invoked first before any other function in this API. this is a one time call, used for doca
   flow initialization and global configurations.
Add one new entry to a pipe.

Parameters

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

Returns

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

Description

When a packet matches a single pipe, will start HW offload. The pipe only defines which fields to match. When offloading, we need detailed information from packets, or we need to set some specific actions that the pipe did not define. The parameters include:
match: The packet detail fields according to the pipe definition. actions: The real actions according to the pipe definition. monitor: Defines the monitor actions if the pipe did not define it. fwd: Define the forward action if the pipe did not define it.

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

__DOCA_EXPERIMENTAL doca_flow_pipe_entry
*doca_flow_pipe_control_add_entry (uint16_t pipe_queue, uint32_t priority, doca_flow_pipe *pipe, const doca_flow_match *match,
const doca_flow_match *match_mask,
const doca_flow_actions *actions, const
doca_flow_action_descs *action_descs, const
doca_flow_monitor *monitor, const doca_flow_fwd *fwd,
doca_flow_error *error)

Add one new entry to a control pipe.

Parameters

pipe_queue
  Queue identifier.

priority
  Priority value.

pipe
  Pointer to pipe.

match
  Pointer to match, indicate specific packet match information.

match_mask
  Pointer to match mask information.

actions
  Pointer to modify actions, indicate specific modify information.

action_descs
  action descriptions

monitor
  Pointer to monitor actions.

fwd
  Pointer to fwd actions.
error
  Output error, set `doca_flow_error` for details.

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

Description
Refer to `doca_flow_pipe_add_entry`.

__DOCA_EXPERIMENTAL__ `doca_flow_pipe`
*`doca_flow_pipe_create (const doca_flow_pipe_cfg *cfg, const doca_flow_fwd *fwd, const doca_flow_fwd *fwd_miss, doca_flow_error *error)`
Create one new pipe.

Parameters

`cfg`
  Pipe configuration.

`fwd`
  Fwd configuration for the pipe.

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

`error`
  Output error, set `doca_flow_error` for details.

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

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

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

This API will create the pipe, but would not start the HW offload.
__DOCA_EXPERIMENTAL void
doca_flow_pipe_destroy (doca_flow_pipe *pipe)
Destroy one pipe.

Parameters
pipe
  Pointer to pipe.

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

__DOCA_EXPERIMENTAL void doca_flow_pipe_dump
(doca_flow_pipe *pipe, FILE *f)
Dump pipe information.

Parameters
pipe
  Pointer to doca flow pipe.
f
  The output file of the pipe information.

doca_flow_entry_status
doca_flow_pipe_entry_get_status
(doca_flow_pipe_entry *entry)
Get entry's status.

Parameters
entry
  pipe entry

Returns
entry's status
**DOCA_EXPERIMENTAL** doca_flow_pipe_entry

*doca_flow_pipe_lpm_add_entry (uint16_t pipe_queue, doca_flow_pipe *pipe, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_fwd *fwd, const doca_flow_flags_type flag, void *usr_ctx, doca_flow_error *error)

Add one new entry to a lpm pipe.

**Parameters**

**pipe_queue**
- Queue identifier.

**pipe**
- Pointer to pipe.

**match**
- Pointer to match, indicate specific packet match information.

**match_mask**
- Pointer to match mask information.

**actions**
- Pointer to modify actions, indicate specific modify information.

**monitor**
- Pointer to monitor actions.

**fwd**
- Pointer to fwd actions.

**flag**
- Flow entry will be pushed to hw immediately or not. enum doca_flow_flags_type.

**usr_ctx**
- Pointer to user context.

**error**
- Output error, set doca_flow_error for details.

**Returns**

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

**Description**

This API will populate the lpm entries.
**doxa_flow_pipe_entry**

*doxa_flow_pipe_ordered_list_add_entry (uint16_t pipe_queue, doxa_flow_pipe *pipe, uint32_t idx, const doxa_flow_ordered_list *ordered_list, const doxa_flow_fwd *fwd, doxa_flow_flags_type flags, void *user_ctx, doxa_flow_error *error)*

**Parameters**

- **pipe_queue**
  - Queue identifier.
- **pipe**
  - Pipe handle.
- **idx**
  - Unique entry index. It is the user’s responsibility to ensure uniqueness.
- **ordered_list**
  - Ordered list with pointers to struct doxa_flow_actions and struct doxa_flow_monitor at the same indices as they were at the pipe creation time. If the configuration contained an element of struct doxa_flow_action_descs, the corresponding array element is ignored and can be NULL.
- **fwd**
  - Entry forward configuration.
- **flags**
  - Entry insertion flags.
- **user_ctx**
  - Opaque context for the completion callback.
- **error**
  - Receives immediate error info.

**Returns**

struct doxa_flow_pipe_entry * The entry inserted.

**Description**

Add an entry to the ordered list pipe.
___DOCA_EXPERIMENTAL int
doca_flow_pipe_rm_entry (uint16_t pipe_queue, void *
usr_ctx, doca_flow_pipe_entry *entry)
Free one pipe entry.

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

Returns
0 on success, negative on failure.

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

___DOCA_EXPERIMENTAL void
doca_flow_port_destroy (doca_flow_port *port)
Destroy a doca port.

Parameters
port
   Pointer to doca flow port.

Description
Destroy the doca port, free all resources of the port.
__DOCA_EXPERIMENTAL int doca_flow_port_pair (doca_flow_port *port, doca_flow_port *pair_port)
pair two doca flow ports.

Parameters

port
  Pointer to doca flow port.

pair_port
  Pointer to the pair port.

Returns

0 on success, negative on failure.

Description

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

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

__DOCA_EXPERIMENTAL void
doca_flow_port_pipes_dump (doca_flow_port *port, FILE *f)
Dump pipe of one port.

Parameters

port
  Pointer to doca flow port.

f
  The output file of the pipe information.

Description

Dump all pipes information belong to this port.
__DOCA_EXPERIMENTAL void
doca_flow_port_pipes_flush (doca_flow_port *port)
Flush pipes of one port.

Parameters
port
    Pointer to doca flow port.

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

__DOCA_EXPERIMENTAL uint8_t *
doca_flow_port_priv_data (doca_flow_port *port)
Get pointer of user private data.

Parameters
port
    Port struct.

Returns
Private data head pointer.

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

__DOCA_EXPERIMENTAL doca_flow_port *
doca_flow_port_start (const doca_flow_port_cfg *
cfg, doca_flow_error *error)
Start a doca port.

Parameters
cfg
    Port configuration, see doca_flow_cfg for details.
error
    Output error, set doca_flow_error for details.
Returns
Port handler on success, NULL otherwise and error is set.

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

__DOCA_EXPERIMENTAL int doca_flow_port_stop (doca_flow_port *port)
Stop a doca port.

Parameters

port
Port struct.

Returns
0 on success, negative on failure.

Description
Stop the port, disable the traffic.

__DOCA_EXPERIMENTAL doca_flow_port *doca_flow_port_switch_get (void)
Get doca flow switch port.

Description
The application could use this function to get the doca switch port, then create pipes and pipe entries on this port.
__DOCA_EXPERIMENTAL int doca_flow_query (doca_flow_pipe_entry *entry, doca_flow_query *query_stats)
Extract information about specific entry.

Parameters

entry
The pipe entry to query.

query_stats
Data retrieved by the query.

Returns
0 on success, negative on failure.

Description
Query the packet statistics about specific pipe entry

__DOCA_EXPERIMENTAL int doca_flow_shared_resource_cfg (doca_flow_shared_resource_type type, uint32_t id, doca_flow_shared_resource_cfg *cfg, doca_flow_error *error)
Configure a single shared resource.

Parameters

type
Shared resource type.

id
Shared resource id.

cfg
Pointer to a shared resource configuration.

error
Output error, set doca_flow_error for details.

Returns
0 on success, negative on failure.
Description
This API can be used by bounded and unbounded resources.

__DOCA_EXPERIMENTAL int
doca_flow_shared_resources_bind
(doca_flow_shared_resource_type type, uint32_t
*res_array, uint32_t res_array_len, void
*bindable_obj, doca_flow_error *error)
Binds a bulk of shared resources to a bindable object.

Parameters

  type
    Shared resource type.
  res_array
    Array of shared resource IDs.
  res_array_len
    Shared resource IDs array length.
  bindable_obj
    Pointer to an allowed bindable object, use NULL to bind globally.
  error
    Output error, set doca_flow_error for details.

Returns
0 on success, negative on failure.

Description
Binds a bulk of shared resources from the same type to a bindable object. Currently the
bindable objects are ports and pipes.
__DOCA_EXPERIMENTAL int
doca_flow_shared_resources_query
(doca_flow_shared_resource_type type, uint32_t
*res_array, doca_flow_shared_resource_result
*query_results_array, uint32_t array_len,
doca_flow_error *error)

Extract information about shared counter.

Parameters

type
    Shared object type.

res_array
    Array of shared objects IDs to query.

query_results_array
    Data array retrieved by the query.

array_len
    Number of objects and their query results in their arrays [same number].

type
    Output error, set doca_flow_error for details.

Returns
0 on success, negative on failure.

Description
Query an array of shared objects of a specific type.

#define DOCA_FLOW_META_EXT 12

External meta data size in bytes.

#define DOCA_FLOW_META_MAX 20

Max meta data size in bytes.

#define DOCA_FLOW_SWITCH
doca_flow_port_switch_get()
Mapping to doca flow switch port.
2.10. **DOCA DMA engine**

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

```c
struct doca_dma_job_memcpy

struct doca_dma_memcpy_result

enum doca_dma_devinfo_caps

Possible DMA device capabilities.

Values

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

enum doca_dma_job_types

Available jobs for DMA.

Values

- `DOCA_DMA_JOB_MEMCPY = DOCA_ACTION_DMA_FIRST+1`

__DOCA_EXPERIMENTAL__ doca_ctx
*doca_dma_as_ctx (doca_dma *dma)

Parameters

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

Returns

Non NULL upon success, NULL otherwise.

Description

Convert doca_dma instance into a generalised context for use with doca core objects.
**doca_error_t doca_dma_create (doca_dma **dma)**

**Parameters**

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

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - dma argument is a NULL pointer.
  - **DOCA_ERROR_NO_MEMORY** - failed to alloc doca_dma.
  - **DOCA_ERROR_INITIALIZATION** - failed to initialise a mutex.

**Description**

Create a DOCA DMA instance.

**doca_error_t doca_dma_destroy (doca_dma *dma)**

**Parameters**

- **dma**
  - Pointer to instance to be destroyed.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_IN_USE** - Unable to gain exclusive access to the dma instance.
  - **DOCA_ERROR_IN_USE** - One or more work queues are still attached. These must be detached first.

**doca_error_t doca_dma_get_max_buf_size (const doca_devinfo *devinfo, uint64_t *buf_size)**

**Parameters**

- **devinfo**
  - The DOCA device information.
- **buf_size**
  - The maximum supported buffer size in bytes.
Returns
DOCA_SUCCESS - upon success Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description
Get the maximum supported buffer size for DMA job.

doca_error_t
doca_dma_get_max_list_buf_num_elem (const
doca_devinfo *devinfo, uint32_t *max_list_num_elem)

Parameters

devinfo
The DOCA device information.

max_list_num_elem
The maximum supported number of elements in a given DOCA linked-list buffer, such that
1 indicates no linked-list buffer support.

Returns
DOCA_SUCCESS - upon success Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.

Description
Get the maximum supported number of elements in a given DOCA linked-list buffer for DMA
job.

doca_error_t doca_dma_job_get_supported
(doca_devinfo *devinfo, doca_dma_job_types
job_type)

Parameters

devinfo
The DOCA device information

job_type
DMA job_type available through this device. see enum doca_dma_job_types.
Returns
DOCA_SUCCESS - in case device supports job_type. Error code - in case of failure:

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

Description
Check if given device is capable of executing a specific DMA job.

2.11. Deep packet inspection

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

`struct doca_dpi_config_t`
DPI init configuration.

`struct doca_dpi_parsing_info`
L2-L4 flow information.

`struct doca_dpi_result`
Dequeue result.

`struct doca_dpi_sig_data`
Extra signature data.

`struct doca_dpi_sig_info`
Signature info.

`struct doca_dpi_stat_info`
DPI statistics.

`enum doca_dpi_dequeue_status_t`
Status of dequeue operation.

Values
- `DOCA_DPI_DEQ_NA`
No DPI enqueued jobs done, or no packets to dequeue

**DOCA_DPI_DEQ READY**  
DPI Job and result is valid

**DOCA_DPI_DEQ NA**  
No DPI enqueued jobs done, or no packets to dequeue

**DOCA_DPI_DEQ READY**  
DPI Job and result is valid

---

**enum doca_dpi_enqueue_status_t**  
Status of enqueue operation.

**Values**

**DOCA_DPI_ENQ_PROCESSING**  
Packet enqueued for processing

**DOCA_DPI_ENQ_PACKET_EMPTY**  
No payload, packet was not queued

**DOCA_DPI_ENQ_BUSY**  
Packet cannot be enqueued, queue is full

**DOCA_DPI_ENQ_INVALID_DB**  
load_signatures failed, or was never called

**DOCA_DPI_ENQ_INTERNAL_ERR**  
Other system errors possible

**DOCA_DPI_ENQ_PROCESSING**  
Packet enqueued for processing

**DOCA_DPI_ENQ_PACKET_EMPTY**  
No payload, packet was not queued

**DOCA_DPI_ENQ_BUSY**  
Packet cannot be enqueued, queue is full

**DOCA_DPI_ENQ_INVALID_DB**  
load_signatures failed, or was never called

**DOCA_DPI_ENQ_INTERNAL_ERR**  
Other system errors possible

---

**enum doca_dpi_flow_status_t**  
Status of enqueued entry.

**Values**

**DOCA_DPI_STATUS_LAST_PACKET = 1<<1**  
Indicates there are no more packets in queue from this flow.

**DOCA_DPI_STATUS_DESTROYED = 1<<2**  
Indicates flow was destroyed while being processed

**DOCA_DPI_STATUS_NEW_MATCH = 1<<3**
Indicates flow was matched on current dequeue

**DOCA_DPI_STATUS_LAST_PACKET = 1<<1**
Indicates there are no more packets in queue from this flow.

**DOCA_DPI_STATUS_DESTROYED = 1<<2**
Indicates flow was destroyed while being processed

**DOCA_DPI_STATUS_NEW_MATCH = 1<<3**
Indicates flow was matched on current dequeue

---

**enum doca_dpi_sig_action_t**

Signature action. Some signatures may come with an action.

**Values**

**DOCA_DPI_SIG_ACTION_NA**
Action not available for signature

**DOCA_DPI_SIG_ACTION_ALERT**
Alert

**DOCA_DPI_SIG_ACTION_PASS**
Signature indicates that the flow is allowed

**DOCA_DPI_SIG_ACTION_DROP**
Signature indicates that the flow should be dropped

**DOCA_DPI_SIG_ACTION_REJECT**
Send RST/ICMP unreach error to the sender of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTSRC**
Send RST/ICMP unreach error to the sender of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTDST**
Send RST/ICMP error packet to receiver of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTBOTH**
Send RST/ICMP error packets to both sides of the conversation

**DOCA_DPI_SIG_ACTION_NA**
Action not available for signature

**DOCA_DPI_SIG_ACTION_ALERT**
Alert

**DOCA_DPI_SIG_ACTION_PASS**
Signature indicates that the flow is allowed

**DOCA_DPI_SIG_ACTION_DROP**
Signature indicates that the flow should be dropped

**DOCA_DPI_SIG_ACTION_REJECT**
Send RST/ICMP unreach error to the sender of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTSRC**
Send RST/ICMP unreach error to the sender of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTDST**
Send RST/ICMP error packet to receiver of the matching packet

**DOCA_DPI_SIG_ACTION_REJECTBOTH**
Send RST/ICMP error packets to both sides of the conversation
Send RST/ICMP error packets to both sides of the conversation

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

Dequeues packets after processing.

**Parameters**

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

**Returns**

doca_dpi_dequeue_status_t if successful, error code otherwise

**Description**

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

```c
__DOCA_EXPERIMENTAL void doca_dpi_destroy
doca_dpi_ctx *ctx)
```

Free the DPI memory and releases the regex engine.

**Parameters**

- **ctx**
  - DPI context to destroy.
Enqueue a new DPI job for processing.

**Parameters**

- `flow_ctx`  
  The flow context handler.

- `pkt`  
  The mbuf to be processed.

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

- `payload_offset`  
  Indicates where the packet’s payload begins.

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

**Returns**

doca_dpi_enqueue_status_t or other error code.

**Description**

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

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

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

- Payload length = 0
__DOCA_EXPERIMENTAL int
doca_dpi_flow_match_get (const doca_dpi_flow_ctx
*flow_ctx, doca_dpi_result *result)
Query a flow’s match.

Parameters
flow_ctx
The flow context of the flow to be queried.
result
Output, latest match on this flow. Only “matched” and “info” fields in the result parameter are valid.

Returns
0 on success, error code otherwise.

__DOCA_EXPERIMENTAL doca_dpi_ctx
*doca_dpi_init (const doca_dpi_config_t *config, int *error)
Initialize the DPI library.

Parameters
config
See doca_dpi_config_t for details.
error
Output error, negative value indicates an error.

Returns
doca_dpi_ctx - dpi opaque context, NULL on error.

Description
This function must be invoked first before any function in the API. It should be invoked once per process. This call will probe the first regex device it finds (0).
__DOCA_EXPERIMENTAL int
doca_dpi_load_signatures (doca_dpi_ctx *ctx, const char *cdo_file)

Loads the cdo file.

Parameters

ctx
  The DPI context.

cdo_file
  CDO file created by the DPI compiler.

Returns

0 on success, error code otherwise.

Description

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

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

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

Returns a specific sig info.

Parameters

ctx
  The DPI context.

sig_id
  The signature ID.

sig_data
  Output of the sig metadata.

Returns

0 on success, error code otherwise.
__DOCA_EXPERIMENTAL int
doca_dpi_signatures_get (const doca_dpi_ctx *ctx,
doca_dpi_sig_data **sig_data)
Returns all signatures.

Parameters

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

Returns

Number of signatures on success, error code otherwise.

Description

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

__DOCA_EXPERIMENTAL void doca_dpi_stat_get
(const doca_dpi_ctx *ctx, bool clear,
doca_dpi_stat_info *stats)
Returns DPI statistics.

Parameters

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

2.12. Remote deep packet inspection (grpc)

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

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

struct doca_dpi_grpc_result
Dequeue result.

struct doca_dpi_parsing_info
L2-L4 flow information.

struct doca_dpi_sig_data
Extra signature data.

struct doca_dpi_sig_info
Signature info.

struct doca_dpi_stat_info
DPI statistics.

enum doca_dpi_dequeue_status_t
Status of dequeue operation.

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

enum doca_dpi_enqueue_status_t
Status of enqueue operation.
Values

DOCA_DPI_ENQ_PROCESSING
Packet enqueued for processing

DOCA_DPI_ENQ_PACKET_EMPTY
No payload, packet was not queued

DOCA_DPI_ENQ_BUSY
Packet cannot be enqueued, queue is full

DOCA_DPI_ENQ_INVALID_DB
load_signatures failed, or was never called

DOCA_DPI_ENQ_INTERNAL_ERR
Other system errors possible

DOCA_DPI_ENQ_PROCESSING
Packet enqueued for processing

DOCA_DPI_ENQ_PACKET_EMPTY
No payload, packet was not queued

DOCA_DPI_ENQ_BUSY
Packet cannot be enqueued, queue is full

DOCA_DPI_ENQ_INVALID_DB
load_signatures failed, or was never called

DOCA_DPI_ENQ_INTERNAL_ERR
Other system errors possible

---

doca_dpi_flow_status_t

Status of enqueued entry.

Values

DOCA_DPI_STATUS_LAST_PACKET = 1<<1
Indicates there are no more packets in queue from this flow.

DOCA_DPI_STATUS_DESTROYED = 1<<2
Indicates flow was destroyed while being processed

DOCA_DPI_STATUS_NEW_MATCH = 1<<3
Indicates flow was matched on current dequeue

DOCA_DPI_STATUS_LAST_PACKET = 1<<1
Indicates there are no more packets in queue from this flow.

DOCA_DPI_STATUS_DESTROYED = 1<<2
Indicates flow was destroyed while being processed

DOCA_DPI_STATUS_NEW_MATCH = 1<<3
Indicates flow was matched on current dequeue

---

doca_dpi_sig_action_t

Signature action. Some signatures may come with an action.
Values

DOCA_DPI_SIG_ACTION_NA
Action not available for signature

DOCA_DPI_SIG_ACTION_ALERT
Alert

DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed

DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped

DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet

DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet

DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet

DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation

DOCA_DPI_SIG_ACTION_NA
Action not available for signature

DOCA_DPI_SIG_ACTION_ALERT
Alert

DOCA_DPI_SIG_ACTION_PASS
Signature indicates that the flow is allowed

DOCA_DPI_SIG_ACTION_DROP
Signature indicates that the flow should be dropped

DOCA_DPI_SIG_ACTION_REJECT
Send RST/ICMP unreach error to the sender of the matching packet

DOCA_DPI_SIG_ACTION_REJECTSRC
Send RST/ICMP unreach error to the sender of the matching packet

DOCA_DPI_SIG_ACTION_REJECTDST
Send RST/ICMP error packet to receiver of the matching packet

DOCA_DPI_SIG_ACTION_REJECTBOTH
Send RST/ICMP error packets to both sides of the conversation
__DOCA_EXPERIMENTAL int doca_dpi_grpc_dequeue (doca_dpi_ctx *ctx, uint16_t dpi_queue, doca_dpi_grpc_result *result)
Dequeues packets after processing.

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

Returns
  doca_dpi_dequeue_status_t if successful, error code otherwise.

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

__DOCA_EXPERIMENTAL void doca_dpi_grpc_destroy (doca_dpi_ctx *ctx)
Close the connection to the DPI service and free the connection memory. This call doesn’t free created flow contexts, make sure to destroy them beforehand.

Parameters
ctx
  The DPI context.
__DOCA_EXPERIMENTAL int doca_dpi_grpc_enqueue
(doca_dpi_flow_ctx *flow_ctx,
doca_dpi_grpc_generic_packet *pkt, bool initiator,
uint32_t payload_offset, void *user_data, size_t
user_data_len, uint16_t dpi_q)
Enqueue a new DPI job for processing.

Parameters
flow_ctx
The flow context handler.

pkt
The packet as binary buffer to be processed.

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

payload_offset
Indicates where the packet’s payload begins.

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

user_data_len
The length of the user_data param.

dpi_q
The DPI queue the flow was created on.

Returns
doca_dpi_enqueue_status_t or other negative error code.

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

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

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

▶ Payload length = 0
__DOCA_EXPERIMENTAL doca_dpi_flow_ctx
*doca_dpi_grpc_flow_create (doca_dpi_ctx *ctx,
uint16_t dpi_q, const doca_dpi_parsing_info
*parsing_info, int *error, doca_dpi_grpc_result
*result)

Creates a new flow on a queue.

Parameters

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

Returns
NULL on error.

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

__DOCA_EXPERIMENTAL void
doca_dpi_grpc_flow_destroy (doca_dpi_flow_ctx *ctx,
uint16_t dpi_q)

Destroys a flow on a queue.

Parameters

ctx
  The flow context to destroy.
dpi_q
  The DPI queue the flow was created on.
Description
Should be called when a flow is terminated or times out

```c
__DOCA_EXPERIMENTAL int
doca_dpi_grpc_flow_match_get (const
doca_dpi_flow_ctx *flow_ctx, doca_dpi_grpc_result
*result, uint16_t dpi_q)
```
Query a flow’s match.

Parameters
- **flow_ctx**
  The flow context of the flow to be queried.
- **result**
  Output, latest match on this flow. Only “matched” and “info” fields in the result parameter are valid.
- **dpi_q**
  The DPI queue the flow was created on.

Returns
0 on success, error code otherwise.

```c
__DOCA_EXPERIMENTAL doca_dpi_ctx
*doca_dpi_grpc_init (const doca_dpi_config_t *config,
int *error)
```
Initialize a connection to the DPI gRPC service.

Parameters
- **config**
  See `doca_dpi_config_t` for details.
- **error**
  Output error, negative value indicates an error.

Returns
doca_dpi_ctx - dpi opaque context, NULL on error.
Description
This function must be invoked first before any function in the API. It should be invoked once per process.

```c
__DOCA_EXPERIMENTAL int
doca_dpi_grpc_load_signatures (doca_dpi_ctx *ctx, const char *path_to_cdo)
```
Loads the cdo file.

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

Returns
0 on success, error code otherwise.

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

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

```c
__DOCA_EXPERIMENTAL int
doca_dpi_grpc_signature_get (const doca_dpi_ctx *ctx, uint32_t sig_id, doca_dpi_sig_data *sig_data)
```
Returns a specific sig info.

Parameters
- **ctx**: The DPI context.
- **sig_id**: The signature ID.
- **sig_data**: Output of the sig metadata.
Returns
0 on success, error code otherwise.

__DOCA_EXPERIMENTAL int
doca_dpi_grpc_signatures_get (const doca_dpi_ctx *ctx, doca_dpi_sig_data **sig_data)
Returns all signatures.

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

Returns
Number of signatures on success, error code otherwise.

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

__DOCA_EXPERIMENTAL void
doca_dpi_grpc_stat_get (const doca_dpi_ctx *ctx, bool clear, doca_dpi_stat_info *stats)
Returns DPI statistics.

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

#define GENERAL_ERRORCODE -1
Unclassified error code for a general error which information is printed to the log.
#define IPV6_ADDER_LEN 16
Length of IPv6 address.

2.13. flow net define

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

enum doca_flow_crypto_action_type
doca flow crypto operation action type

Values

DOCA_FLOW_CRYPTO_ACTION_NONE = 0
No crypto action performed
DOCA_FLOW_CRYPTO_ACTION_ENCRYPT
Perform encryption
DOCA_FLOW_CRYPTO_ACTION_DECRYPT
Perform decryption/authentication

enum doca_flow_crypto_header_type
doca flow crypto operation encapsulation header type

Values

DOCA_FLOW_CRYPTO_HEADER_NONE = 0
No network header involved
DOCA_FLOW_CRYPTO_HEADER_IPV4
IPv4 network header type
DOCA_FLOW_CRYPTO_HEADER_IPV6
IPv6 network header type
DOCA_FLOW_CRYPTO_HEADER_IPV4_UDP
IPv4 + UDP network header type
DOCA_FLOW_CRYPTO_HEADER_IPV6_UDP
IPv6 + UDP network header type

enum doca_flow_crypto_net_type
doca flow crypto operation network mode type

Values

DOCA_FLOW_CRYPTO_NET_NONE = 0
No network header involved

**DOCA_FLOW_CRYPTO_NET_TUNNEL**
Tunnel network header

**DOCA_FLOW_CRYPTO_NET_TRANSPORT**
Transport network header

**enum doca_flow_crypto_protocol_type**
doca flow crypto operation protocol type

**Values**

- **DOCA_FLOW_CRYPTO_PROTOCOL_NONE = 0**
  No security protocol engaged
- **DOCA_FLOW_CRYPTO_PROTOCOL_NISP**
  NISP protocol action
- **DOCA_FLOW_CRYPTO_PROTOCOL_ESP_DECRYPT = 100**
  IPsec ESP protocol decrypt action
- **DOCA_FLOW_CRYPTO_PROTOCOL_ESP_ENCRYPT = 101**
  IPsec ESP protocol encrypt action

**enum doca_flow_crypto_reformat_type**
doca flow crypto operation reformat type

**Values**

- **DOCA_FLOW_CRYPTO_REFORMAT_NONE = 0**
  No reformat action performed
- **DOCA_FLOW_CRYPTO_REFORMAT_ENCAP**
  Perform encapsulation action
- **DOCA_FLOW_CRYPTO_REFORMAT_DECAP**
  Perform decapsulation action

## 2.14. Flow

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

struct doca_flow_grpc_fwd
forwarding configuration wrapper

struct doca_flow_grpc_pipe_cfg
pipeline configuration wrapper

struct doca_flow_grpc_response
General DOCA Flow response struct.

c enum doca_flow_grpc_bindable_obj_type
doca flow grpc bindable object types

Values

DOCA_FLOW_GRPC_BIND_TYPE_PIPE
bind resource to a pipe

DOCA_FLOW_GRPC_BIND_TYPE_PORT
bind resource to a port

DOCA_FLOW_GRPC_BIND_TYPE_NULL
bind resource globally

doca_flow_grpc_aging_handle (uint16_t port_id,
uint16_t queue, uint64_t quota, uint64_t *entries_id,
int len)
RPC call for doca_flow_aging_handle().

Parameters

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

__DOCA_EXPERIMENTAL void
doca_flow_grpc_client_create (const char *grpc_address)
Initialize a channel to DOCA flow grpc server.

Parameters

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

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

__DOCA_EXPERIMENTAL void
doca_flow_grpc_destroy (void)
RPC call for doca_flow_destroy().

doca_flow_grpc_entries_process (uint16_t port_id, uint16_t pipe_queue, uint64_t timeout, uint32_t max_processed_entries)
RPC call for doca_flow_grpc_entries_process().

Parameters

port_id
Port ID

pipe_queue
Queue identifier.

timeout
Max time in micro seconds for this function to process entries. Process once if timeout is 0

max_processed_entries
Flow entries number to process If it is 0, it will proceed until timeout.
Returns
doca_flow_grpc_response

doca_flow_grpc_init (const doca_flow_cfg *cfg)
RPC call for doca_flow_init().

Parameters
cfg
    Program configuration, see doca_flow_cfg for details.

Returns
doca_flow_grpc_response.

doca_flow_grpc_pipe_add_entry (uint16_t pipe_queue, uint64_t pipe_id, const doca_flow_match *match, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_grpc_fwd *client_fwd, uint32_t flags)
RPC call for doca_flow_pipe_add_entry().

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

Returns
doca_flow_grpc_response.
doaca_flow_grpc_pipe_control_add_entry (uint16_t pipe_queue, uint8_t priority, uint64_t pipe_id, const doaca_flow_match *match, const doaca_flow_match *match_mask, const doaca_flow_grpc_fwd *client_fwd)

RPC call for doaca_flow_pipe_control_add_entry().

Parameters

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

Returns
doca_flow_grpc_response.

doca_flow_grpc_pipe_create (const doaca_flow_grpc_pipe_cfg *cfg, const doaca_flow_grpc_fwd *fwd, const doaca_flow_grpc_fwd *fwd_miss)

RPC call for doaca_flow_pipe_create().

Parameters

cfg
  Pipe configuration, see doaca_flow_grpc_pipe_cfg for details.
fwd
  Fwd configuration for the pipe.
**fwd_miss**

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

Returns

doca_flow_grpc_response.

doca_flow_grpc_pipe_destroy (uint64_t pipe_id)

RPC call for doca_flow_pipe_destroy().

Parameters

**pipe_id**

Pipe ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_pipe_entry_get_status (uint64_t entry_id)

RPC call for doca_flow_pipe_entry_get_status().

Parameters

**entry_id**

pipe entry ID.

Returns

doca_flow_grpc_response
```c
#define doca_flow_grpc_pipe_lpm_add_entry (uint16_t pipe_queue, uint64_t pipe_id, const doca_flow_match *match, const doca_flow_match *match_mask, const doca_flow_actions *actions, const doca_flow_monitor *monitor, const doca_flow_grpc_fwd *client_fwd, const doca_flow_flags_type flag)
```

RPC call for `doca_flow_pipe_lpm_add_entry()`.

**Parameters**

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

**Returns**

- `doca_flow_grpc_response`

```c
#define doca_flow_grpc_pipe_rm_entry (uint16_t pipe_queue, uint64_t entry_id)
```

RPC call for `doca_flow_grpc_pipe_rm_entry()`.

**Parameters**

- `pipe_queue` 
  Queue identifier.
- `entry_id` 
  The entry ID to be removed.
Returns

doca_flow_grpc_response.

doca_flow_grpc_port_destroy (uint16_t port_id)
RPC call for doca_flow_port_destroy().

Parameters

port_id
Port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_pair (uint16_t port_id, uint16_t pair_port_id)
RPC call for doca_flow_port_pair().

Parameters

port_id
port ID.

pair_port_id
pair port ID.

Returns

doca_flow_grpc_response.

doca_flow_grpc_port_pipes_dump (uint16_t port_id, FILE *f)
RPC call for doca_flow_port_pipes_dump().

Parameters

port_id
Port ID.

f
The output file of the pipe information.

Returns

doca_flow_grpc_response.
**doxa_flow_grpc_port_pipes_flush (uint16_t port_id)**

RPC call for `doxa_flow_port_pipes_flush()`.

**Parameters**

- **port_id**
  - Port ID.

**Returns**

`doxa_flow_grpc_response`.

**doxa_flow_grpc_port_start (const doxa_flow_port_cfg *cfg)**

RPC call for `doxa_flow_port_start()`.

**Parameters**

- **cfg**
  - Port configuration, see `doxa_flow_port_cfg` for details.

**Returns**

`doxa_flow_grpc_response`.

**doxa_flow_grpc_port_stop (uint16_t port_id)**

RPC call for `doxa_flow_port_stop()`.

**Parameters**

- **port_id**
  - Port ID.

**Returns**

`doxa_flow_grpc_response`.

**doxa_flow_grpc_port_switch_get (void)**

RPC call for `doxa_flow_port_switch_get()`.

**Returns**

`doxa_flow_grpc_response`
doxa_flow_grpc_query (uint64_t entry_id, 
doca_flow_query *query_stats)
RPC call for doca_flow_query().

Parameters

entry_id
The pipe entry ID to query.

query_stats
Data retrieved by the query.

Returns
doca_flow_grpc_response.

doca_flow_grpc_shared_resource_cfg
(doca_flow_shared_resource_type type, uint32_t id, 
doca_flow_shared_resource_cfg *cfg)
RPC call for doca_flow_shared_resource_cfg().

Parameters

type
Shared resource type.

id
Shared resource id.

cfg
Pointer to a shared resource configuration.

Returns
doca_flow_grpc_response.
**doса_flow_grpc_shared_resources_bind**
(dоса_flow_shаrd_resource_type type, uint32_t *res_array, uint32_t res_array_len, doса_flow_grpc_bindаle_obj *bindаle_obj_id)

RPC call for docа_flow_shаrd_resources_bind().

**Parameters**

- **type**  
  Shared resource type.

- **res_array**  
  Array of shared resource IDs.

- **res_array_len**  
  Shared resource IDs array length.

- **bindаle_obj_id**  
  Pointer to a bindаle object ID.

**Returns**

dоса_flow_grpc_response.

**doса_flow_grpc_shared_resources_query**
(dоса_flow_shаrd_resource_type type, uint32_t *res_array, doса_flow_shаrd_resource_result *query_results_array, uint32_t array_len)

RPC call for docа_flow_shаrd_resources_query().

**Parameters**

- **type**  
  Shared object type.

- **res_array**  
  Array of shared objects IDs to query.

- **query_results_array**  
  Data array retrieved by the query.

- **array_len**  
  Number of objects and their query results in their arrays (same number).

**Returns**

0 on success, negative on failure.
2.15. flow net define

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

struct doca_flow_ip_addr
doca flow ip address

struct doca_flow_tun
doca flow tunnel information

enum doca_flow_ip_type
doca flow ip address type

Values

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

enum doca_flow_tun_type
doca flow tunnel type

Values

DOCA_FLOW_TUN_NONE = 0
  tunnel is not set
DOCA_FLOW_TUN_VXLAN
  tunnel is vxlan type
DOCA_FLOW_TUN_GTPU
  tunnel is gtpu type
DOCA_FLOW_TUN_GRE
  tunnel is gre type
DOCA_FLOW_TUN_NISP
  tunnel is nisp type
DOCA_FLOW_TUN_AUDP
  tunnel is nisp type
DOCA_FLOW_TUN_ESP
  tunnel is nisp type
typedef uint16_t doca_be16_t
16-bit big-endian value.

typedef uint32_t doca_be32_t
32-bit big-endian value.

typedef uint64_t doca_be64_t
64-bit big-endian value.

#define DOCA_ETHER_ADDR_LEN (6)
length of ether add length.

#define DOCA_ETHER_TYPE_IPV4 (0x0800)
Ethernet frame types IPv4 Protocol.

#define DOCA_ETHER_TYPE_IPV6 (0x86DD)
IPv6 Protocol.

#define DOCA_ETHER_TYPE_TEB (0x6558)
Transparent Ethernet Bridging.

#define DOCA_FLOW_AUDP_DWORD 6
AUDP header maximal length in dwords

#define DOCA_FLOW_AUDP_HEADER_LEN (DOCA_FLOW_AUDP_DWORD * sizeof(doca_be32_t))
AUDP header maximal length in bytes

#define DOCA_FLOW_CRYPTO_KEY_LEN_MAX 32
Crypto key maximal length in bytes
#define DOCA_FLOW_CRYPTO_REFORMAT_LEN_MAX
(DOCA_ETHER_ADDR_LEN * 2 + sizeof(doca_be16_t)
+ sizeof(doca_be16_t) * 2 * 2 + sizeof(doca_be32_t)
* 15 + sizeof(doca_be32_t) * 2 +
DOCA_FLOW_NISP_HEADER_LEN)

NISP/ESP tunnel header may consist of:

- Ethernet addresses
- Ethernet type
- optional VLAN and 802.1Q headers
- IPv4 (with full options) or IPv6 (w/o options)
- optional UDP header
- NISP or ESP header

#define DOCA_FLOW_ESP_HEADER_LEN (2 *
sizeof(doca_be32_t))

IPsec ESP header maximal length in bytes

#define DOCA_FLOW_NISP_DWORD 10

NISP header maximal length in dwords

#define DOCA_FLOW_NISP_HEADER_LEN
(DOCA_FLOW_NISP_DWORD * sizeof(doca_be32_t))

NISP header maximal length in bytes

#define DOCA_GTPU_PORT (2152)

gtpu upd port id.

#define DOCA_NISP_DEFAULT_PORT (1000)

default nisp/audp port id.
#define DOCA_PROTO_GRE (47)
Cisco GRE tunnels (rfc 1701,1702).

#define DOCA_PROTO_TCP (6)
Transmission Control Protocol.

#define DOCA_PROTO_UDP (17)
User Datagram Protocol.

#define DOCA_VXLAN_DEFAULT_PORT (4789)
default vxlan port id.

2.16. IPsec
DOCA IPSEC library. For more details please refer to the user guide on DOCA devzone.

struct doca_encryption_key
IPSec encryption key.

struct doca_ipsec_sa_antireplay
IPSec antireplay attributes, part of ipsec attr.

struct doca_ipsec_sa_attr
IPSec attributes to create jobs.

struct doca_ipsec_sa_create_job
DOCA IPSec SA creation job.

struct doca_ipsec_sa_destroy_job
DOCA IPSec SA destroy job.

enum doca_encryption_key_type
IPSec encryption key type.
Values

DOCA_ENCRYPTION_KEY_AESGCM_128
  size of 128 bit
DOCA_ENCRYPTION_KEY_AESGCM_256
  size of 256 bit

enum doca_ipsec_direction
IPSec direction of the key, incoming packets or outgoing.

Values

DOCA_IPSEC_DIRECTION_INGRESS_Decrypt = 0
  incoming packets, decryption
DOCA_IPSEC_DIRECTION_EGRESS_Encrypt = 1
  outgoing packets, encryption

enum doca_ipsec_icv_length
IPSec icv length.

Values

DOCA_IPSEC_ICV_LENGTH_8 = 8
  size of 8 bit
DOCA_IPSEC_ICV_LENGTH_12 = 12
  size of 12 bit
DOCA_IPSEC_ICV_LENGTH_16 = 16
  size of 16 bit

enum doca_ipsec_job_types
Docs ipsec action type enums, used to specify ipsec job types.

Values

DOCA_IPSEC_JOB_SA_CREATE = DOCA_ACTION_IPSEC_FIRST+1
  create sa object
DOCA_IPSEC_JOB_SA_DESTROY
  destroy sa object

enum doca_ipsec_replay_win_size
IPSec replay window size.

Values

DOCA_IPSEC_REPLAY_WIN_SIZE_32 = 32
size of 32 bit
DOCA_IPSEC_REPLAY_WIN_SIZE_64 = 64
size of 64 bit
DOCA_IPSEC_REPLAY_WIN_SIZE_128 = 128
size of 128 bit
DOCA_IPSEC_REPLAY_WIN_SIZE_256 = 256
size of 256 bit

enum doca_ipsec_sa_mode
IPSec protocol mode.

Values
DOCA_IPSEC_SA_MODE_TRANSPORT = 1
   IPSec Transport mode
DOCA_IPSEC_SA_MODE_TUNNEL
   IPSec Tunnel mode

enum doca_ipsec_sa_offload
IPSec offload mode.

Values
DOCA_IPSEC_SA_OFFLOAD_FULL = 1
   IPSec full offload
DOCA_IPSEC_SA_OFFLOAD_CRYPTO
   IPSec ipsec offload

enum doca_ipsec_sa_protocol
IPSec protocol.

Values
DOCA_IPSEC_SA_PROTO_ESP = 1
   ESP protocol
DOCA_IPSEC_SA_PROTO_AH
   AH protocol unsupported, added for consistency
__DOCA_EXPERIMENTAL doca_ctx
*doca_ipsec_as_ctx (doca_ipsec *ctx)
Convert IPSec instance into context for use with workQ.

Parameters
ctx
IPSEC instance. This must remain valid until after the context is no longer required.

Returns
Non NULL - doca_ctx object on success. Error:
> NULL.

doca_error_t doca_ipsec_create (doca_ipsec **ctx)
Create a DOCA IPSEC instance.

doca_error_t doca_ipsec_destroy (doca_ipsec *ctx)
Destroy DOCA IPSEC instance.

Parameters
ctx
Instance to be destroyed, MUST NOT BE NULL.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
> DOCA_ERROR_INVALID_VALUE - received invalid input.
> DOCA_ERROR_IN_USE - the ctx still in use by one or more workQs.

doca_error_t doca_ipsec_is_mode_supported (const
doca_devinfo *devinfo, doca_ipsec_sa_mode mode)
Get is mode supported [reffer to doca_ipsec_sa_attrs.mode].

Parameters
devinfo
The DOCA device information
mode
The mode to query the capability
Returns
DOCA_SUCCESS - in case of success - capability supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given capability.

doca_error_t doca_ipsec_is_offload_supported (const doca_devinfo *devinfo, doca_ipsec_sa_offload offload)
Get is mode offload supported [reffer to doca_ipsec_sa_attrs.offload].

Parameters

- devinfo
  The DOCA device information
- offload
  The offload to query the capability

Returns
DOCA_SUCCESS - in case of success - capability supported. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given capability.

doca_error_t doca_ipsec_is_protocol_supported (const doca_devinfo *devinfo, doca_ipsec_sa_protocol protocol)
Get is mode protocol ESP [reffer to doca_ipsec_sa_attrs.protocol].

Parameters

- devinfo
  The DOCA device information
- protocol
  The protocol to query the capability

Returns
DOCA_SUCCESS - in case of success - capability supported. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given capabilitie.

**doca_error_t doca_ipsec_job_get_supported**

*(doca_devinfo *devinfo, doca_ipsec_job_types job_type)*

Check if given device is capable for given doca_ipsec job.

**Parameters**

- **devinfo**
  - The DOCA device information
- **job_type**
  - doca_ipsec job type. See enum doca_ipsec_job_types.

**Returns**

DOCA_SUCCESS - in case the job is supported. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_ipsec job.

**__DOCA_EXPERIMENTAL doca_ipsec_sa**

*doca_ipsec_sa_from_result (doca_event *ev)*

Convert IPSec event job into sa object.

**Parameters**

- **ev**
  - event of ipsec job

**Returns**

Non NULL - sa object of ipsec. Error:

- NULL.

### 2.17. Logging Management

Define functions for internal and external logging management.

To add DOCA internal logging compile with “-D DOCA_LOGGING_ALLOW_DLOG”
**doca_log_registrator**

Registers log source on program start.

cppClassifierVisibility: visibility=public

**enum doca_log_level**

log levels

**Values**

- **DOCA_LOG_LEVEL_CRIT = 20**
  
  Critical log level

- **DOCA_LOG_LEVEL_ERROR = 30**
  
  Error log level

- **DOCA_LOG_LEVEL_WARNING = 40**
  
  Warning log level

- **DOCA_LOG_LEVEL_INFO = 50**
  
  Info log level

- **DOCA_LOG_LEVEL_DEBUG = 60**
  
  Debug log level

**typedef (*log_flush_callback) (char* buffer)**

logging backend flush() handler

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

Generates a log message.

**Parameters**

- **level**
  
  Log level enum DOCA_LOG_LEVEL.

- **source**
  
  The log source identifier defined by doca_log_source_register.

- **line**
  
  The line number this log originated from.

- **format**
  
  printf(3) arguments, format and variables.

**Returns**

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

doca_error_t doca_log_backend_level_set
doca_logger_backend *logger, uint32_t level)
Set the log level of a specific logger backend.

Parameters

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

Returns
DOCA error code.

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


doca_error_t doca_log_create_buffer_backend (char
*buffer, size_t capacity, log_flush_callback handler,
doca_logger_backend **backend)
Create a logging backend with a char buffer stream.

Parameters

- **buffer**
  - The char buffer (char *) for the logger’s stream.
- **capacity**
  - Maximal amount of chars that could be written to the stream.
- **handler**
  - Handler to be called when the log record should be flushed from the stream.
- **backend**
  - Logging backend that wraps the given buffer [only valid if no error occurred].
Returns
DOCA error code.

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

doca_error_t doca_log_create_fd_backend (int fd, doca_logger_backend **backend)
Create a logging backend with an fd stream.

Parameters
fd
The file descriptor (int) for the logger’s backend.
backend
Logging backend that wraps the given fd (only valid if no error occurred).

Returns
DOCA error code.

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

doca_error_t doca_log_create_file_backend (FILE *fptr, doca_logger_backend **backend)
Create a logging backend with a FILE* stream.

Parameters
fptr
The FILE * for the logger’s stream.
backend
Logging backend that wraps the given fptr (only valid if no error occurred).

Returns
DOCA error code.
Description
Creates a new logging backend that will be added on top of the default logger.

doca_error_t doca_log_create_syslog_backend (const char *name, doca_logger_backend **backend)
Create a logging backend with a syslog output.

Parameters
- **name**
  The syslog name for the logger’s backend.
- **backend**
  Logging backend that exposes the desired syslog functionality (only valid if no error occurred).

Returns
DOCA error code.

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

doca_error_t __DOCA_EXPERIMENTAL
doca_log_developer (uint32_t level, int source, int line, const char *format, ...)
Generates a log message for DLOG operations.

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

Returns
DOCA error code.
Description
The log will be shown in the doca_log_stream_redirect (see default).

Note:
This function is thread safe.

__DOCA_EXPERIMENTAL uint16_t
doca_log_get_bucket_time (void)
Get the timespan of the rate-limit bucket.

Returns
Time [in seconds] of the rate-limit bucket.

__DOCA_EXPERIMENTAL uint16_t
doca_log_get_quantity (void)
Get the quantity of the rate-limit bucket.

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

__DOCA_EXPERIMENTAL uint32_t
doca_log_global_level_get (void)
Get the log level of the default logger backend.

Returns
Log level enum DOCA_LOG_LEVEL.

Description
Dynamically query for the log level of the default logger backend, any log under this level will be shown.
**doca_error_t doca_log_global_level_set (uint32_t level)**

Set the log level of the default logger backend.

**Parameters**

- **level**
  - Log level enum DOCA_LOG_LEVEL.

**Returns**

DOCA error code.

**Description**

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

**doca_error_t doca_log_rate_bucket_register (int source, int *bucket)**

Register a new rate bucket.

**Parameters**

- **source**
  - The log source identifier defined by doca_log_source_register.
- **bucket**
  - Bucket identifier that was allocated to this log source [only valid if no error occurred].

**Returns**

DOCA error code.

**Description**

Will return the identifier associated with the new bucket.
__DOCA_EXPERIMENTAL void doca_log_rate_limit (uint32_t level, int source, int line, int bucket, const char *format, ...)
Generates a log message with rate limit.

Parameters

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

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

__DOCA_EXPERIMENTAL void doca_log_set_bucket_time (uint16_t bucket_time)
Set the timespan of the rate-limit bucket.

Parameters

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

__DOCA_EXPERIMENTAL void doca_log_set_quantity (uint16_t quantity)
Set the quantity of the rate-limit bucket.

Parameters

- **quantity**
  Maximal number of log events for a rate-limit bucket.
**doca_error_t doca_log_source_destroy (int source)**

Destroy a log source.

**Parameters**

**source**

The source identifier of source to be destroyed, as allocated by `doca_log_source_register`.

**Returns**

DOCA error code.

**Description**

Destroys a given log source as part of the teardown process of the running program.

**Note:**

Used automatically via DOCA_LOG_REGISTER, not recommended to call it directly.

---

**doca_error_t doca_log_source_register (const char *source_name, int *source)**

Register a log source.

**Parameters**

**source_name**

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

**source**

Source identifier that was allocated to this log source name (only valid if no error occurred).

**Returns**

DOCA error code.

**Description**

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

**Note:**

Recommended to only be used via DOCA_LOG_REGISTER.
doca_error_t doca_log_stream_redirect (FILE *stream)

Redirect the logger to a different stream.

Parameters

stream
   Pointer to the stream.

Returns

DOCA error code.

Description

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

#define DOCA_DLOG do { \ } while (0)

Generates a development log message.

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

#define DOCA_DLOG_CRIT DOCA_DLOG(CRIT, format, ##__VA_ARGS__)

Generates a CRITICAL development log message.

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

#define DOCA_DLOGDBG DOCA_DLOG(DBG, format, ##__VA_ARGS__)

Generates a DEBUG development log message.

Will generate debug log for development purposes. To show the logs define DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.
#define DOCA_DLOG_ERR DOCA_DLOG(ERROR, format, ##__VA_ARGS__)
Generates an ERROR development log message.
Will generate error log for development purposes. To show the logs define
DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance
if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_INFO DOCA_DLOG(INFO, format, ##__VA_ARGS__)
Generates an INFO development log message.
Will generate info log for development purposes. To show the logs define
DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance
if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_DLOG_WARN DOCA_DLOG(WARNING, format, ##__VA_ARGS__)
Generates a WARNING development log message.
Will generate warning log for development purposes. To show the logs define
DOCA_LOGGING_ALLOW_DLOG in the compilation variables. This will not effect performance
if compiled without DOCA_LOGGING_ALLOW_DLOG, as it will be removed by the compiler.

#define DOCA_LOG

doca_log(DOCA_LOG_LEVEL_##level, log_source, __LINE__, format, ##__VA_ARGS__)
Generates a log message.
The DOCA_LOG is the main log function for logging. This call affects the performance.
Consider using DOCA_DLOG for the option to remove it on the final compilation. Consider
using the specific level DOCA_LOG for better code readability (i.e. DOCA_LOG_ERR).
#define DOCA_LOG_CRIT DOCA_LOG(CRIT, format, ##__VA_ARGS__)
Generates a CRITICAL log message.
Will generate critical log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_DBG DOCA_LOG(DEBUG, format, ##__VA_ARGS__)
Generates a DEBUG log message.
Will generate debug log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_ERR DOCA_LOG(ERROR, format, ##__VA_ARGS__)
Generates an ERROR log message.
Will generate error log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.

#define DOCA_LOG_INFO DOCA_LOG(INFO, format, ##__VA_ARGS__)
Generates an INFO log message.
Will generate info log. This call affects the performance. Consider using DOCA_DLOG for the option to remove it on the final compilation.
#define DOCA_LOG_RATE_LIMIT do { \ static 
   int log_bucket = -1; \ if (log_bucket == -1) { \ 
   doca_log_rate_bucket_register(log_source, 
   &log_bucket); \ } \ 
   doca_log_rate_limit(DOCA_LOG_LEVEL_##level, 
   log_source, __LINE__, log_bucket, format, 
   ##__VA_ARGS__); \ } while (0)
Generates a log message with rate limit.

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

#define DOCA_LOG_RATE_LIMIT_CRIT
DOCA_LOG_RATE_LIMIT(CRIT, format, 
##__VA_ARGS__) 
Generates a CRITICAL rate limited log message.

#define DOCA_LOG_RATE_LIMIT_DBG
DOCA_LOG_RATE_LIMIT(DBG, format, 
##__VA_ARGS__) 
Generates a DEBUG rate limited log message.

#define DOCA_LOG_RATE_LIMIT_ERR
DOCA_LOG_RATE_LIMIT(ERR, format, 
##__VA_ARGS__) 
Generates an ERROR rate limited log message.

#define DOCA_LOG_RATE_LIMIT_INFO
DOCA_LOG_RATE_LIMIT(INFO, format, 
##__VA_ARGS__) 
Generates an INFO rate limited log message.
#define DOCA_LOG_RATE_LIMIT_WARN
DOCA_LOG_RATE_LIMIT(WARNING, format, ##__VA_ARGS__)

Generates a WARNING rate limited log message.

#define DOCA_LOG_WARN DOCA_LOG(WARNING, format, ##__VA_ARGS__)

Generates a WARNING log message.

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

2.18. RegEx engine

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

struct doca_regex_job_search

struct doca_regex_match

struct doca_regex_search_result

enum doca_regex_job_types

Available job types for RegEx.

Values

DOCA_REGEX_JOB_SEARCH = DOCA_ACTION_REGEX_FIRST+1

Default RegEx search mode

enum doca_regex_search_job_flags

Available job flags for RegEx.

Values

DOCA_REGEX_SEARCH_JOB_FLAG_HIGHEST_PRIORITY_MATCH = 1<<1
DOCA_REGEX_SEARCH_JOB_FLAG_STOP_ON_ANY_MATCH = 1<<2
enum doca_regex_status_flag

Response status flags

Values

DOCA_REGEX_STATUS_SEARCH_FAILED = 1

__DOCA_EXPERIMENTAL doca_ctx
*doca_regex_as_ctx (doca_regex *regex)

Parameters

regex
The RegEx instance to convert. This must remain valid until after the context is no longer required.

Returns

Non NULL upon success, NULL otherwise.

Description

Convert RegEx instance into context for use with workQ

doca_error_t doca_regex_create (doca_regex **regex)

Parameters

regex
Pointer to be populated with the address of the newly created RegEx context.

Returns

DOCA_SUCCESS - RegEx instance was created
DOCA_ERROR_INVALID_VALUE - received invalid input.
DOCA_ERROR_NO_MEMORY - Unable to create required resources.

Description

Create a DOCA RegEx instance.
doca_error_t doca_regex_destroy (doca_regex *regex)

Parameters
regex
   Instance to be destroyed, MUST NOT BE NULL.

Returns
DOCA_SUCCESS - RegEx instance was created DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Destroy DOCA RegEx instance.

Note:
The context must be stopped via a successful call to doca_ctx_stop before it can be safely destroyed.

doca_error_t
doca_regex_get_failed_job_fallback_enabled (const doca_regex *regex, bool *enabled)

Parameters
regex
   The RegEx engine.
enabled
   Set to true when the feature is enabled, or false if it is disabled.

Returns
DOCA_SUCCESS - enabled is populated DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Is the failed job fallback feature enabled.
doca_error_t
doca_regex_get_hardware_compiled_rules (const
doca_regex *regex, void **rules_data, size_t
*rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently
stored. Caller is assumes ownership of this memory and can choose to release it at any
time.

rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

Returns

DOCA_SUCCESS - rules_data and rules_data_size are populated.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable
to allocate memory to store a copy of the rules.

Description

Get the compiled rules data to be used by the hardware RegEx device.

doca_error_t
doca_regex_get_hardware_supported
(const doca_devinfo *devinfo)

Parameters

devinfo
Device to check.

Returns

DOCA_SUCCESS - Hardware acceleration is supported. DOCA_ERROR_INVALID_VALUE -
received invalid input. DOCA_ERROR_NOT_SUPPORTED - Hardware acceleration is NOT
supported.

Description

Determine if a given device supports hardware accelerated RegEx searches.
**doca_error_t**

**doca_regex_get_hardware_uncompiled_rules**

`const doca_regex *regex, void **rules_data, size_t *rules_data_size`

**Parameters**

- **regex**
  The RegEx engine.

- **rules_data**
  Value to populate to hold a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.

- **rules_data_size**
  Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.

**Returns**

- **DOCA_SUCCESS** - rules_data and rules_data_size are populated.
- **DOCA_ERROR_INVALID_VALUE** - received invalid input. **DOCA_ERROR_NO_MEMORY** - Unable to allocate memory to store a copy of the rules.

**Description**

Get the un-compiled rules data to be used by the hardware RegEx device.

**doca_error_t**

**doca_regex_get_huge_job_emulation_overlap_size**

`const doca_regex *regex, uint16_t *nb_overlap_bytes`

**Parameters**

- **regex**
  The RegEx engine.

- **nb_overlap_bytes**
  Number of bytes of overlap in use.
Returns
DOCA_SUCCESS - nb_overlap_bytes is populated. DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Get the size of overlap to use when a job exceeds a device's maximum search size.

doca_error_t doca_regex_get_maximum_job_size (const doca_devinfo *devinfo, uint64_t *max_job_len)

Parameters

devinfo
  Device to check.
max_job_len
  Maximum supported job length.

Returns
DOCA_SUCCESS - max_job_len is populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device does not support RegEx.

Description
Determine the maximum job size supported by this device.

doca_error_t
doca_regex_get_maximum_non_huge_job_size (const doca_devinfo *devinfo, uint64_t *max_job_len)

Parameters

devinfo
  Device to check.
max_job_len
  Maximum supported job length.

Returns
DOCA_SUCCESS - max_job_len is populated. DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device does not support RegEx.
Description
Determine the maximum job size supported by this device without requiring the huge job emulation feature.

doca_error_t
doca_regex_get_small_job_offload_threshold (const
doca_regex *regex, uint16_t *threshold)

Parameters
regex
The RegEx engine.
threshold

Returns
DOCA_SUCCESS - threshold is populated. DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Get the "small jobs" threshold.

doca_error_t
doca_regex_get_software_compiled_rules (const
doca_regex *regex, void **rules_data, size_t
*rules_data_size)

Parameters
regex
The RegEx engine.

rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently stored. Caller is assumes ownership of this memory and can choose to release it at any time.

rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.
Returns
DOCA_SUCCESS - rules_data and rules_data_size are populated.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable
to allocate memory to store a copy of the rules.

Description
Get the compiled rules data to be used by the software RegEx device.

doca_error_t doca_regex_get_software_supported
(const doca_devinfo *devinfo)

Parameters
devinfo
Device to check.

Returns
DOCA_SUCCESS - Software support is available. DOCA_ERROR_INVALID_VALUE - received
invalid input. DOCA_ERROR_NOT_SUPPORTED - Software support is NOT available.

Description
Determine if a given device supports software based RegEx searches.

doca_error_t
doca_regex_get_software_uncompiled_rules
(const doca_regex *regex, void **rules_data, size_t *rules_data_size)

Parameters
regex
The RegEx engine.

rules_data
Value to populate with a pointer to an array of bytes which are a copy of the value currently
stored. Caller is assumes ownership of this memory and can choose to release it at any
time.

rules_data_size
Size of the array pointed to by rules_data. Only valid when *rules_data != NULL.
Returns
DOCA_SUCCESS - rules_data and rules_data_size are populated.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_MEMORY - Unable
to allocate memory to store a copy of the rules.

Description
Get the un-compiled rules data to be used by the software RegEx device.

doca_error_t
doca_regex_get_workq_matches_memory_pool_size
(const doca_regex *regex, uint32_t *pool_size)

Parameters
regex
    The RegEx engine.
pool_size
    Number of items that will be available to each workq.

Returns
DOCA_SUCCESS - pool_size is populated. DOCA_ERROR_INVALID_VALUE - received invalid
input.

Description
Get the size of work queue pool attached to workq for use with the RegEx engine.

doca_error_t doca_regex_is_supported (const
doca_devinfo *devinfo)

Parameters
devinfo
    Device to check.

Returns
DOCA_SUCCESS - Device can be used with doca_regex. DOCA_ERROR_INVALID_VALUE
- received invalid input. DOCA_ERROR_NOT_SUPPORTED - Device cannot be used with
doca_regex.
Description
Determine if a given device is suitable for use with doca_regex.

doca_error_t doca_regex_job_get_supported (const
doca_devinfo *devinfo, doca_regex_job_types
job_type)

Parameters
devinfo
   Device to check.
job_type
   Job type to check.

Returns
DOCA_SUCCESS - Job type is supported with the given device. DOCA_ERROR_INVALID_VALUE
- received invalid input. DOCA_ERROR_NOT_SUPPORTED - Job type is NOT supported with
the given device.

Description
Determine if the given job type is supported for the given device.

doca_error_t
doca_regex_search_job_flag_get_highest_priority_match_supported
(const doca_devinfo *devinfo)

Parameters
devinfo
   Device to check.

Returns
DOCA_SUCCESS - Search jobs support highest priority match when using the given device.
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NOT_SUPPORTED -
highest priority match is NOT supported when using the given device.
**Description**

Determine if ‘highest priority’ match is supported for the given device when submitting `doca_regex_job_search` jobs.

```c
**doca_error_t**
doca_regex_search_job_flag_get_stop_on_any_match_supported
    (const doca_devinfo *devinfo)
```

**Parameters**

- `devinfo`
  - Device to check.

**Returns**

- `DOCA_SUCCESS` - Search jobs support stop on any match when using the given device.
- `DOCA_ERROR_INVALID_VALUE` - received invalid input. `DOCA_ERROR_NOT_SUPPORTED` - stop on any match is NOT supported when using the given device.

**Description**

Determine if ‘stop on any’ match is supported for the given device when submitting `doca_regex_job_search` jobs.

```c
**doca_error_t**
doca_regex_set_failed_job_fallback_enabled
    (doca_regex *regex, bool enabled)
```

**Parameters**

- `regex`
  - The RegEx engine.
- `enabled`
  - Specify true to enable the feature, false to disable it.

**Returns**

- `DOCA_SUCCESS` - Property was successfully set
- `DOCA_ERROR_INVALID_VALUE` - received invalid input.
- `DOCA_ERROR_NO_LOCK` - Unable to gain exclusive control of RegEx instance.
- `DOCA_ERROR_IN_USE` - RegEx instance is currently started.
Description

Enable the ability to automatically migrate a job which was executed on the hardware device and subsequently failed to be re-executed on the software device. This is useful if a hardware limitation prevents a job from executing to completion.

Note:
This feature requires both a hardware and a software device to be available. Validated during context start.

doca_error_t
doca_regex_set_hardware_compiled_rules
doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters

regex
The RegEx engine.

rules_data
An opaque blob of rules data which is provided to the hardware device.

rules_data_size
Size of the blob.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the compiled rules data to be used by the hardware RegEx device.

Note:

- This property is mutually exclusive with hardware un-compiled rules. This property mandates that a hardware device will be attached before the context is started. If no hardware device will be provided you should not specify hardware rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context.
The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.

doca_error_t
doca_regex_set_hardware_uncompiled_rules
doca_regex_set_hardware_uncompiled_rules
(docu_regex *regex, const void *rules_data, size_t
   rules_data_size)

Parameters

regex
   The RegEx engine.

rules_data
   An opaque blob of rules data which will be compiled by the engine into compiled rules data which is then provided to the hardware device.

rules_data_size
   Size of the blob.

Returns

DOCA_SUCCESS - Property was successfully set Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description

Specify the un-compiled rules data to be used by the hardware RegEx device.

Note:

- This property is mutually exclusive with hardware compiled rules. This property mandates that a hardware device will be attached before the context is started. If no hardware device will be provided you should not specify hardware rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context
- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.
- Rules compilation takes place during context start.
**dooca_error_t**  
**dooca_regex_set huge_job_emulation_overlap_size**  
*(dooca_regex *regex, uint16_t nb_overlap_bytes)*

**Parameters**

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

**Returns**

- DOCA_SUCCESS - Property was successfully set  
- Error code - in case of failure:  
  - DOCA_ERROR_INVALID_VALUE - received invalid input.  
  - DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance.  
  - DOCA_ERROR_IN_USE - RegEx instance is currently started.

**Description**

Set the size of overlap to use when a job exceeds a devices maximum search size. Defaults to 0 (no overlap)

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

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

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

**Note:**

The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context.
doxa_error_t
doca_regex_set_small_job_offload_threshold
(doca_regex *regex, uint16_t threshold)

Parameters
regex
   The RegEx engine.
threshold
   Threshold job size in bytes.

Returns
DOCA_SUCCESS - Property was successfully set
Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input.
DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance.
DOCA_ERROR_IN_USE - RegEx instance is currently started.

Description
Define a threshold for “small jobs”. For scenarios where small jobs cause poor performance using the hardware RegEx device these can instead be redirected to the software device. Set this to a value > 0 to enable the feature. Set this value to 0 to disable the feature. Defaults to 0 (disabled)

Note:
This feature requires both a hardware and a software device to be available. The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context

doxa_error_t
doca_regex_set_software_compiled_rules
(doca_regex *regex, const void *rules_data, size_t rules_data_size)

Parameters
regex
   The RegEx engine.
**rules_data**
An opaque blob of rules data which is provided to the software device.

**rules_data_size**
Size of the blob.

**Returns**

DOCA_SUCCESS - Property was successfully set
Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input.
DOCA_ERROR_NO_LOCK - Unable to gain exclusive control of RegEx instance.
DOCA_ERROR_IN_USE - RegEx instance is currently started.
DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

**Description**

Specify the compiled rules data to be used by the software RegEx device.

**Note:**
- This property is mutually exclusive with software un-compiled rules. This property mandates that a software device will be attached before the context is started. If no software device will be provided you should not specify software rules, or explicitly clear them by setting them to NULL. This will be validated as part of starting the context.
- The caller retains ownership of data pointed to by rules_data and is responsible for freeing it when they no longer require it. The engine will make a copy of this data for its own purposes.

**doca_error_t**

**doca_regex_set_software_uncompiled_rules**

(docapapgy *regex, const void *rules_data, size_t rules_data_size)

**Parameters**

**regex**
The RegEx engine.

**rules_data**
An opaque blob of rules data which will be compiled by the engine into compiled rules data which is then provided to the software device.

**rules_data_size**
Size of the blob
Returns
DOCA_SUCCESS - Property was successfully set Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to
gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently
started. DOCA_ERROR_NO_MEMORY - Unable to allocate memory to store a copy of the rules.

Description
Specify the un-compiled rules data to be used by the software RegEx device.

Note:
- This property is mutually exclusive with software compiled rules. This property mandates
  that a software device will be attached before the context is started. If no software device
  will be provided you should not specify software rules, or explicitly clear them by setting
  them to NULL. This will be validated as part of starting the context
- The caller retains ownership of data pointed to by rules_data and is responsible for freeing
  it when they no longer require it. The engine will make a copy of this data for its own
  purposes.
- Rules compilation takes place during context start.

doca_error_t
doca_regex_set_workq_matches_memory_pool_size
(docaregex *regex, uint32_t pool_size)

Parameters
regex
The RegEx engine.
pool_size
Number of items to have available to each workq.

Returns
DOCA_SUCCESS - Property was successfully set Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input. DOCA_ERROR_NO_LOCK - Unable to
gain exclusive control of RegEx instance. DOCA_ERROR_IN_USE - RegEx instance is currently
started.
Description

Each work queue attached to the RegEx engine gets a pool allocator for matches. Set this value to set the maximum number of matches that can be stored for a given workq.

Note:
The range of valid values for this property depend upon the device in use. This means that acceptance of a value through this API does not ensure the value is acceptable, this will be validated as part of starting the context.

2.19. RegEx engine memory pool

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

__DOCA_EXPERIMENTAL doca_regex_mempool
*doca_regex_mempool_create (size_t elem_size,
size_t nb_elems)

Parameters

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

Returns

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

Description

Create a memory pool.

Note:
Supports single producer and single consumer only.
__DOCA_EXPERIMENTAL void
doca_regex_mempool_destroy
(doca_regex_mempool *pool)

Parameters

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

Description
Destroy a memory pool and all objects it owned.

Note:
all pointers to elements in this pool must be cleared before this call. Failure to do so may result in undefined behaviour.

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

Parameters

pool
Memory pool to fetch an object from.

n
Index of the object to be retrieved

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

Description
Directly access an object in the mempool by index.

Note:
- this function does not care if the object is in use or free.
- Supports single producer and single consumer only.
__DOCA_EXPERIMENTAL void
*doca_regex_mempool_get_obj
(doca_regex_mempool *pool)

Parameters

pool
Pool from which to get a free object.

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

Description
Get an object from the memory pool.

Note:
Supports single producer and single consumer only.

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

Parameters

pool
Memory pool owning the object.

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

Returns
0 based index of element or a negative error code.

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

Note:
Supports single producer and single consumer only.
__DOCA EXPERIMENTAL void
doca_regex_mempool_put_obj
(doca_regex_mempool *pool, void *obj)

Parameters

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

Description

Put an object back into the memory pool.

Note:
Supports single producer and single consumer only.

2.20. DOCA RMAX engine

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

struct doca_rmax_cpu_affinity_mask
Data structure to describe CPU mask for doca_rmax internal thread.

struct doca_rmax_in_stream_completion
Completion returned by input stream describing the incoming packets.

struct doca_rmax_stream_error
Detailed completion error information.

enum doca_rmax_action_type
action types for doca jobs

Values

\[ \text{DOCA\_RMAX\_ACTION\_TYPE\_RX\_DATA} = \text{DOCA\_ACTION\_RMAX\_FIRST+1} \]
  Input packets data. This action does not originate from a submitted job.
enum doca_rmax_in_stream_scatter_type
Incoming packet scatter mode, used by input stream.

Values

DOCA_RMAX_IN_STREAM_SCATTER_TYPE_RAW = 0
Store raw packet data including network headers

DOCA_RMAX_IN_STREAM_SCATTER_TYPE_ULP
Store User-Level Protocol only data (discard network header up to L4)

DOCA_RMAX_IN_STREAM_SCATTER_TYPE_PAYLOAD
Store payload data only (all headers will be discarded)

enum doca_rmax_in_stream_ts_fmt_type
Input packet timestamp format (timestamp, when packet was received).

Values

DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_COUNTER = 0
Raw number written by HW, representing the HW clock

DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_NANO
Time in nanoseconds

DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_SYNCED
Time in nanoseconds, synced with PTP grandmaster

enum doca_rmax_in_stream_type
Type of input stream.

Values

DOCA_RMAX_IN_STREAM_TYPE_GENERIC = 0
Generic stream

DOCA_RMAX_IN_STREAM_TYPE_RTP_2110
SMPTE ST 2110 stream

typedef uint64_t doca_rmax_cpu_mask_t
CPU bitmask container
`doca_error_t doca_rmax_flow_attach (const doca_rmax_flow *flow, const doca_rmax_in_stream *stream)`
Attach a flow to a stream.

**Parameters**
- **flow**
  - Flow to operate on
- **stream**
  - The context for attaching a flow

**Returns**
- `DOCA_SUCCESS` - in case of success.
- `doca_error` code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - received invalid input.
  - `DOCA_ERROR_INITIALIZATION` - Rivermax is not initialized.
  - `DOCA_ERROR_SHUTDOWN` - library shutdown in a process.
  - `DOCA_ERROR_UNEXPECTED` - unexpected issue.

`doca_error_t doca_rmax_flow_create (doca_rmax_flow **flow)`
Create a steering flow for input stream to filter incoming data flow by match criteria.

**Parameters**
- **flow**
  - The flow created for input stream. Non NULL upon success, NULL otherwise.

**Returns**
- `DOCA_SUCCESS` - in case of success.
- `doca_error` code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - received invalid input.
  - `DOCA_ERROR_NO_MEMORY` - unable to allocate memory.
**doxa_error_t doca_rmax_flow_destroy** *(doxa_rmax_flow *flow)*

Destroy a steering flow.

**Parameters**

- **flow**
  Flow to destroy.

**Returns**

- DOCA_SUCCESS - in case of success.
- doca_error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.

**doxa_error_t doca_rmax_flow_detach** *(const doxa_rmax_flow *flow, const doxa_rmax_in_stream *stream)*

Detach a flow from a stream.

**Parameters**

- **flow**
  Flow to operate on
- **stream**
  The context for detaching a flow

**Returns**

- DOCA_SUCCESS - in case of success.
- doca_error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
  - DOCA_ERROR_SHUTDOWN - library shutdown in a process.
  - DOCA_ERROR_UNEXPECTED - unexpected issue.

**doxa_error_t doca_rmax_flow_set_dst_ip** *(doxa_rmax_flow *flow, const in_addr *ip)*

Set the destination IP filter for the flow.

**Parameters**

- **flow**
  Flow to operate on
ip
   Destination IPv4 address

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
‣ DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_dst_port
(doca_rmax_flow *flow, uint16_t port)
Set the destination port filter for the flow.

Parameters
flow
   Flow to operate on
port
   Destination port number, non-zero

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
‣ DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t doca_rmax_flow_set_src_ip
(doca_rmax_flow *flow, const in_addr *ip)
Set the source IP filter for the flow.

Parameters
flow
   Flow to operate on
ip
   Source IPv4 address

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
‣ DOCA_ERROR_INVALID_VALUE - received invalid input.
**doca_error_t doca_rmax_flow_set_src_port**(doca_rmax_flow *flow, uint16_t port)

Set the source port filter for the flow.

**Parameters**

- **flow**
  - Flow to operate on
- **port**
  - Source port number. If zero then any source port is accepted.

**Returns**

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.

**doca_error_t doca_rmax_flow_set_tag**(doca_rmax_flow *flow, uint32_t tag)

Set the tag for the flow.

**Parameters**

- **flow**
  - Flow to operate on
- **tag**
  - Non-zero tag

**Returns**

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.

**doca_error_t doca_rmax_get_cpu_affinity_mask**(doca_rmax_cpu_affinity_mask *mask)

Get affinity mask for the internal Rivermax thread.

**Parameters**

- **mask**
  - Affinity mask. CPU is included in affinity mask if the corresponding bit is set. If CPU affinity mask is unset return value is zeroed.
DOCA error_t doca_rmax_get_ptp_clock_supported (const doca_devinfo *devinfo)

Query PTP clock capability for device.

Parameters

devinfo
   The device to query

Returns

DOCA_SUCCESS - PTP clock is supported. DOCA_ERROR_NOT_SUPPORTED - PTP clock is not supported. doca_error code - in case of failure:
   ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
   ▶ DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
   ▶ DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv4 address from the OS

__DOCA_EXPERIMENTAL doca_ctx *
doca_rmax_in_stream_as_ctx
(doca_rmax_in_stream *stream)

Convert a DOCA RMAX input stream to DOCA context.

Parameters

stream
   The context to be converted

Returns

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

Description

DOCA RMAX stream supports all stream operations: create/start/stop/destroy. Only one device and one workq must be attached to a stream.
doca_error_t doca_rmax_in_stream_create (doca_rmax_in_stream **stream)
Create a DOCA RMAX input stream context.

Parameters
stream
The input stream context created for the DOCA RMAX. Non NULL upon success, NULL otherwise.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - stream argument is a NULL pointer.
- DOCA_ERROR_NO_MEMORY - failed to alloc doca_rmax_in_stream.
- DOCA_ERROR_INITIALIZATION - failed to initialise DOCA context.

Description
Create input stream.

doca_error_t doca_rmax_in_stream_destroy (doca_rmax_in_stream *stream)
Destroy a DOCA input stream context.

Parameters
stream
The context to be destroyed

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - stream argument is a NULL pointer.

Description
Free all allocated resources associated with a DOCA RMAX input stream.
**doca_error_t**

`doca_rmax_in_stream_get_elements_count (const doca_rmax_in_stream *stream, uint32_t *value)`

Get number of elements in the stream buffer.

**Parameters**

- **stream**
  The input stream to query.
- **value**
  Where to write the current property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error** code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.

**Description**

This value can differ from value set by `doca_rmax_in_stream_set_elements_count` if the argument is not a power of two.

**doca_error_t**

`doca_rmax_in_stream_get_max_packets (const doca_rmax_in_stream *stream, uint32_t *value)`

Get maximal number of packets that input stream must return in read event.

**Parameters**

- **stream**
  The input stream to query.
- **value**
  Where to write the current property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error** code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.
**docta_error_t**

`docta_rmax_in_stream_get_memblk_size (const docta_rmax_in_stream *stream, size_t *value)`

Get size of memory block[s].

**Parameters**

- **stream**
  - The input stream to query.

- **value**
  - Where to write the current property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.
  - **DOCA_ERROR_INITIALIZATION** - Rivermax is not initialized.
  - **DOCA_ERROR_UNEXPECTED** - unexpected program flow.

**Description**

Size of memory block [array of sizes for multiple memblks, the number of memory blocks in stream is more than one].

**docta_error_t**

`docta_rmax_in_stream_get_memblk_stride_size (const docta_rmax_in_stream *stream, uint16_t *value)`

Get stride size[s].

**Parameters**

- **stream**
  - The input stream to query.

- **value**
  - Where to write the current property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.
- DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

**Description**

Stride size of memory block (array of stride sizes for multiple memory blocks).

```c
#include <doca.h>

doca_error_t
doca_rmax_in_stream_get_memblks_count (const
doca_rmax_in_stream *stream, uint32_t *value)
```

Get number of configured memory blocks.

**Parameters**

- `stream`  
  The input stream to query.
- `value`  
  Where to write the current property value.

**Returns**

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.

**Description**

Amount of memblks is equal to the number of segments into which the incoming packet will be divided.

```c
#include <doca.h>

doca_error_t
doca_rmax_in_stream_get_min_packets (const
doca_rmax_in_stream *stream, uint32_t *value)
```

Get minimal number of packets that input stream must return in read event.

**Parameters**

- `stream`  
  The input stream to query.
- `value`  
  Where to write the current property value.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.

`doca_error_t`

doca_rmax_in_stream_get_scatter_type
(const doca_rmax_in_stream *stream, doca_rmax_in_stream_scatter_type * value)

Get the type of packet’s data scatter.

**Parameters**

- `stream`
  The input stream to query.

- `value`
  Where to write the current property value.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

**Description**

See enum `doca_rmax_in_stream_scatter_type`.

`doca_error_t doca_rmax_in_stream_get_timeout_us`

(const doca_rmax_in_stream *stream, int *value)

Get receive timeout.

**Parameters**

- `stream`
  The input stream to query.

- `value`
  Where to write the current property value.

**Returns**

Description

The number of usecs that library would do busy wait (polling) for reception of at least `min_packets` number of packets.

doca_error_t
doca_rmax_in_stream_get_timestamp_format
(const doca_rmax_in_stream *stream,
doca_rmax_in_stream_ts_fmt_type * *value)
Get stream timestamp format.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description

See enum doca_rmax_in_stream_ts_fmt_type

doca_error_t doca_rmax_in_stream_get_type
(const doca_rmax_in_stream *stream,
doca_rmax_in_stream_type * *value)
Get input stream type.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - received invalid input.

doca_error_t
doca_rmax_in_stream_memblk_desc_get_max_size
(const doca_rmax_in_stream *stream, uint16_t *value)
Get maximal packet segment sizes.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

Description

Array of maximal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer.

doca_error_t
doca_rmax_in_stream_memblk_desc_get_min_size
(const doca_rmax_in_stream *stream, uint16_t *value)
Get minimal packet segment sizes.

Parameters

stream
The input stream to query.

value
Where to write the current property value.

Returns

DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

**Description**

Array of minimal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer.

```c
typedef void doca_error_t;

doca_error_t
doca_rmax_in_stream_memblk_desc_set_max_size(
doca_rmax_in_stream *stream, const uint16_t *value)
```

Set maximal packet segment sizes.

**Parameters**

- `stream`  
  The input stream to write property.

- `value`  
  Property value.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

**Description**

Array of maximal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer. Must be set before starting the stream context.
**doca_error_t**

**doca_rmax_in_stream_memblk_desc_set_min_size**

*(doca_rmax_in_stream *stream, const uint16_t *value)*

Set minimal packet segment sizes.

**Parameters**

- **stream**
  The input stream to write property.

- **value**
  Property value.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - unexpected program flow.

**Description**

Array of minimal packet segment sizes that will be received by input stream. Array length equals to the number of memory blocks in the stream buffer. Default: 0.

**doca_error_t**

**doca_rmax_in_stream_set_elements_count**

*(doca_rmax_in_stream *stream, uint32_t value)*

Set number of elements in the stream buffer.

**Parameters**

- **stream**
  The input stream to write property.

- **value**
  Property value.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
**Description**

Must be set before starting the stream context. See also `doca_rmax_in_stream_get_elements_count`.

```c

doca_error_t
doca_rmax_in_stream_set_max_packets
(doca_rmax_in_stream *stream, uint32_t value)
```

Set maximal number of packets that input stream must return in read event.

**Parameters**

- **stream**
  - The input stream to write property.
- **value**
  - Property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.

**Description**

Default: 1024.

```c

doca_error_t
doca_rmax_in_stream_set_memblk
(doca_rmax_in_stream *stream, doca_buf *buf)
```

Set memory buffer(s).

**Parameters**

- **stream**
  - The input stream to write property.
- **buf**
  - Property value.

**Returns**

- **DOCA_SUCCESS** - in case of success.
- **doca_error code** - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - received invalid input.
  - **DOCA_ERROR_UNEXPECTED** - unexpected program flow.
Description
Memory buffer (or head of linked list of memory buffers) for storing received data. The length of linked list must be the same as number of memory blocks configured. Must be set before starting the stream context.

doca_error_t
doca_rmax_in_stream_set_memblks_count
(doc_a_rmax_in_stream *stream, uint32_t value)
Set number of configured memory blocks.

Parameters
- **stream**
  The input stream to write property.
- **value**
  Property value.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
- **DOCA_ERROR_INVALID_VALUE** - received invalid input.

Description
Amount of memblks is equal to the number of segments into which the incoming packet will be divided. Default: 1. Valid values: 1 and 2.

doca_error_t
doca_rmax_in_stream_set_min_packets
(doc_a_rmax_in_stream *stream, uint32_t value)
Set minimal number of packets that input stream must return in read event.

Parameters
- **stream**
  The input stream to write property.
- **value**
  Property value.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Default: 0.

doca_error_t doca_rmax_in_stream_set_scatter_type
(doca_rmax_in_stream *stream,
doca_rmax_in_stream_scatter_type value)
Set the type of packet’s data scatter.

Parameters
stream
   The input stream to write property.
value
   Property value.

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
See enum doca_rmax_in_stream_scatter_type. Default:
DOCA_RMAX_IN_STREAM_SCATTER_TYPE_RAW.

doca_error_t doca_rmax_in_stream_set_timeout_us
(doca_rmax_in_stream *stream, int value)
Set receive timeout.

Parameters
stream
   The input stream to write property.
value
   Property value.
Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
The number of usecs that library would do busy wait (polling) for reception of at least `min_packets` number of packets.
Default: 0.

doca_error_t
doca_rmax_in_stream_set_timestamp_format
(doca_rmax_in_stream *stream,
doca_rmax_in_stream_ts_fmt_type value)
Set stream timestamp format.

Parameters
  stream
  The input stream to write property.
  value
  Property value.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
See enum doca_rmax_in_stream_ts_fmt_type Default:
DOCA_RMAX_IN_STREAM_TS_FMT_TYPE_RAW_COUNTER.
**doca_error_t** **doca_rmax_in_stream_set_type**(doca_rmax_in_stream *stream, doca_rmax_in_stream_type value)

Set input stream type.

**Parameters**

- **stream**
  The input stream to write property.

- **value**
  Property value.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.

**Description**

Default: DOCA_RMAX_IN_STREAM_TYPE_GENERIC.

**doca_error_t** **doca_rmax_init**(void)

DOCA RMAX library initialization.

**Returns**

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

- DOCA_ERROR_UNSUPPORTED_VERSION - unsupported Rivermax library version.
- DOCA_ERROR_INITIALIZATION - Rivermax initialization failed.
- DOCA_ERROR_NO_MEMORY - unable to allocate memory.
- DOCA_ERROR_NOT_FOUND - there are no supported devices.
- DOCA_ERROR_NOT_SUPPORTED - invalid or missing Rivermax license.
- DOCA_ERROR_UNEXPECTED - unexpected issue.

**Description**

This function initializes the DOCA RMAX global resources. This function must be called after **doca_rmax_set_cpu_affinity_mask** and before any other DOCA RMAX library call.
__DOCA_EXPERIMENTAL void doca_rmax_interrupt (void)
Interrupt the currently executing DOCA RMAX function, if any.

doca_error_t doca_rmax_release (void)
Uninitialize DOCA RMAX library.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INITIALIZATION - Rivermax is not initialized.
  ▶ DOCA_ERROR_IN_USE - library is in use.

Description
This function cleans up the DOCA RMAX resources. No DOCA RMAX function may be called after calling this function.

doca_error_t doca_rmax_set_clock (doca_dev *dev)
Set the device to use for obtaining PTP time.

Parameters
dev
  Device to use for obtaining the PTP time.

Returns
DOCA_SUCCESS - in case of success. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
  ▶ DOCA_ERROR_NOT_FOUND - there is no IPv4 address associated with this device.
  ▶ DOCA_ERROR_NOT_SUPPORTED - PTP clock is not supported by device.
  ▶ DOCA_ERROR_OPERATING_SYSTEM - failed to acquire the IPv4 address from the OS
  ▶ DOCA_ERROR_UNEXPECTED - unexpected issue.

Description
The device must have PTP clock capability, see doca_rmax_get_ptp_clock_supported.
**doca_error_t doca_rmax_set_cpu_affinity_mask**

**(const doca_rmax_cpu_affinity_mask *mask)**

Set affinity mask for the internal Rivermax thread.

**Parameters**

**mask**

Affinity mask. CPU is included in affinity mask if the corresponding bit is set. By default affinity mask is not set, so internal thread can run on any CPU core.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - invalid affinity mask provided.

**Description**

Must be called before doca_rmax_init().

```c
#define DOCA_RMAX_CPU_SETSIZE 1024
```

maximum CPU set size

```c
#define DOCA_RMAX_NCPUBITS (8 * sizeof(doca_rmax_cpu_mask_t))
```

number of CPU bits per one cpu mask element

### 2.21. engine

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

**struct doca_sha_job**

**struct doca_sha_partial_job**

**enum doca_sha_job_flags**

Flag enum for SHA job.
Values

DOCA_SHA_JOB_FLAGS_NONE = 0
    Default flag for all SHA job.
DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL
    Only useful for doca_sha_partial_job.

enum doca_sha_job_type

Doca sha action type enums, used to specify sha job types.

Values

DOCA_SHA_JOB_SHA1 = DOCA_ACTION_SHA_FIRST+1
DOCA_SHA_JOB_SHA256
DOCA_SHA_JOB_SHA512
DOCA_SHA_JOB_SHA1_PARTIAL
DOCA_SHA_JOB_SHA256_PARTIAL
DOCA_SHA_JOB_SHA512_PARTIAL

__DOCA_EXPERIMENTAL doca_ctx *doca_sha_as_ctx (doca_sha *ctx)

Parameters

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

Returns

Non NULL - doca_ctx object on success. Error:

> NULL.

Description

Convert sha instance into context for use with workQ

doca_error_t doca_sha_create (doca_sha **ctx)

Parameters

ctx
    Instance pointer to be created, MUST NOT BE NULL.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
  ▶ DOCA_ERROR_NO_MEMORY - not enough memory for allocation.
  ▶ DOCA_ERROR_NOT_SUPPORTED - the required engine is not supported.

Description
Create a DOCA SHA instance.

doca_error_t doca_sha_destroy (doca_sha *ctx)

Parameters
  ctx
  Instance to be destroyed, MUST NOT BE NULL.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
  ▶ DOCA_ERROR_IN_USE - the required engine resource is not released yet. Please call
doca_ctx_stop()  

Description
Destroy DOCA SHA instance.

doca_error_t doca_sha_get_hardware_supported (const doca_devinfo *devinfo)

Parameters
  devinfo
  The DOCA device information

Returns
DOCA_SUCCESS - in case of success, it is a hardware_based engine. doca_error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - received invalid input.
DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description
Get DOCA SHA operating mode: hardware or openssl_fallback. First to use `doca_sha_job_get_supported()` to decide whether the devinfo support doca_sha engine. Second to use `doca_sha_get_hardware_supported()` to decide whether this doca_sha engine is hardware_based or openssl_fallback_based.

doca_error_t doca.sha.get_max_list_buf_num_elem(const doca_devinfo *devinfo, uint32_t *max_list_num_elem)

Parameters

devinfo
The DOCA device information.

max_list_num_elem
The maximum supported number of elements in a given DOCA linked-list buffer, such that 1 indicates no linked-list buffer support.

Returns
DOCA_SUCCESS - upon success Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - in case of invalid input.
- DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities. or provided devinfo does not support the given doca_sha job.

Description
Get the maximum supported number of elements in a given DOCA linked-list buffer for SHA job.
**doct_error_t doca_sha_get_max_src_buffer_size**  
(const doca_devinfo *devinfo, uint64_t *max_buffer_size)

**Parameters**
- **devinfo**: The DOCA device information  
- **max_buffer_size**: The max buffer size for DOCA SHA operation in bytes.

**Returns**
- DOCA_SUCCESS - upon success  
- Error code - in case of failure:  
  - DOCA_ERROR_INVALID_VALUE - in case of invalid input.  
  - DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_sha job.

**Description**
Get maximum source buffer size for DOCA SHA job.

**doct_error_t doca_sha_get_min_dst_buffer_size**  
(const doca_devinfo *devinfo, doca_sha_job_type job_type, uint32_t *min_buffer_size)

**Parameters**
- **devinfo**: The DOCA device information  
- **job_type**: doca_sha job type. See enum doca_sha_job_type.  
- **min_buffer_size**: The min buffer size for DOCA SHA operation in bytes.

**Returns**
- DOCA_SUCCESS - in case of success  
- doct_error code - in case of failure:  
  - DOCA_ERROR_INVALID_VALUE - received invalid input.  
  - DOCA_ERROR_NOT_SUPPORTED - failed to query device capabilities or provided devinfo does not support the given doca_sha job.
**Description**

Get minimum result buffer size for DOCA SHA job.

```c
#include <doctx.h>

int doca_sha_job_get_supported(const doca_devinfo *devinfo, doca_sha_job_type job_type)
```

**Parameters**

- **devinfo**
  - The DOCA device information
- **job_type**
  - SHA job_type available through this device. see enum doca_sha_job_type.

**Returns**

- DOCA_SUCCESS - in case device supports job_type.
- Error code - in case of failure:
  - DOCA_ERROR_INVALID_VALUE - received invalid input.
  - DOCA_ERROR_NOT_SUPPORTED - provided devinfo does not support this SHA job.

**Description**

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

```c
#include <doctx.h>

doca_error_t doca_sha_partial_session_copy(doca_sha *ctx, doca_workq *workq, const doca_sha_partial_session *from_session, doca_sha_partial_session *to_session)
```

**Parameters**

- **ctx**
  - SHA instance.
- **workq**
  - Workq instance.
- **from_session**
  - The source sha_partial_session object to be copied.
- **to_session**
  - The dest sha_partial_session object.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_UNEXPECTED - received corrupted input.

Description
Copy the sha_partial session state from "from_session" to "to_session". This is useful if large amounts of data to be sha-calculated which share the same header segments, and only differ in the tail last few bytes. For example, we have two msgs: msg_0 = {header, tail_0}, msg_1 = {header, tail_1}, both of them need to be processed in sha_partial job.

when without session_copy functionality: we will do sha_calculation for msg_0 as:
```
doca_sha_partial_session_create(ctx, workq, &session_0);
doca_workq_submit(workq, &job_0_for_header);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_workq_submit(workq, &job_0_for_tail_0);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_sha_partial_session_destroy(ctx, workq, session_0);
```
then, we will do sha_calculation for msg_1 as:
```
doca_sha_partial_session_create(ctx, workq, &session_1);
doca_workq_submit(workq, &job_1_for_header);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_workq_submit(workq, &job_1_for_tail_1);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_sha_partial_session_destroy(ctx, workq, session_1);
```
Obviously, the same data "header" is calculated twice!

when utilising the session_copy functionality, we will do sha_calculation for msg_0 as:
```
doca_sha_partial_session_create(ctx, workq, &session_0);
doca_workq_submit(workq, &job_0_for_header);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_sha_partial_session_copy(ctx, workq, session_0, session_1);
```
do a session_copy:
```
doca_sha_partial_session_create(ctx, workq, &session_1);
doca_sha_partial_session_copy(ctx, workq, session_0, session_1);
```
continue to finish msg_0:
```
doca_workq_submit(workq, &job_0_for_tail_0);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_sha_partial_session_destroy(ctx, workq, session_0);
```
continue to finish msg_1:
```
doca_workq_submit(workq, &job_1_for_tail_1);
doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);
doca_sha_partial_session_destroy(ctx, workq, session_1);
```
**doxa_error_t doca_sha_partial_session_create**

```c
(doca_sha *ctx, doca_workq *workq,
doca_sha_partial_session **session)
```

**Parameters**

- **ctx**
  SHA instance.
- **workq**
  Workq instance.
- **session**
  Instance pointer to be created, MUST NOT BE NULL.

**Returns**

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

- DOCA_ERROR_INVALID_VALUE - received invalid input.
- DOCA_ERROR_NO_MEMORY - the resource is exhausted for now, please wait the existing jobs to finish or call doca_sha_partial_session_destroy() to release the allocated session objects.

**Description**

Get a session object used for sha_partial calculation. For the same sha_partial job, user need to keep using the same sha_partial_session. A session object can only be used for the same sha_ctx and workq. It cannot be shared between different sha_ctx or different workq.

**doxa_error_t doca_sha_partial_session_destroy**

```c
(doca_sha *ctx, doca_workq *workq,
doca_sha_partial_session *session)
```

**Parameters**

- **ctx**
  SHA instance.
- **workq**
  Workq instance.
- **session**
  A sha_partial_session object to be released.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

▶ DOCA_ERROR_INVALID_VALUE - received invalid input.

Description
Release the session object used for sha_partial calculation. Please make sure the session to be released belonging to the specific sha_ctx and workq.

#define DOCA_SHA1_BYTE_COUNT 20
User response buffer length should be >= the following specified length, depending on the SHA function. Both SHA1_PARTIAL and SHA1 need 20 bytes. Both SHA256_PARTIAL and SHA256 need 32 bytes. Both SHA512_PARTIAL and SHA512 need 64 bytes. SHA1 calculation hex-format result length.

#define DOCA_SHA256_BYTE_COUNT 32
SHA256 calculation hex-format result length.

#define DOCA_SHA512_BYTE_COUNT 64
SHA512 calculation hex-format result length.

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

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

Limitations:
The lib supports the netflow V9 format. The lib is not thread safe.
enum doca_telemetry_ipc_status_t
DOCA telemetry IPC status.

Values
DOCA_TELEMETRY_IPC_STATUS_FAILED = -1
DOCA_TELEMETRY_IPC_STATUS_CONNECTED
DOCA_TELEMETRY_IPC_STATUS_DISABLED

typedef uint8_t doca_guid_t
DOCA GUID type.

typedef uint64_t doca_telemetry_timestamp_t
DOCA schema type index type.

typedef uint8_t doca_telemetry_type_index_t
DOCA schema field type index.

doca_error_t doca_telemetry_check_ipc_status
(doca_telemetry_source *doca_source,
doca_telemetry_ipc_status_t *status)
Return status of IPC transport.

Parameters
doca_source
Input doca source.
status
DOCA_TELEMETRY_IPC_STATUS_FAILED - if IPC is not connected.
DOCA_TELEMETRY_IPC_STATUS_CONNECTED - if IPC is connected.
DOCA_TELEMETRY_IPC_STATUS_DISABLED - if IPC is disabled from config.
    if return is DOCA_SUCCESS then status can be one of the following

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
▶ DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.
▶ DOCA_TELEMETRY_IPC_STATUS_FAILED - if IPC is not connected.
▶ DOCA_TELEMETRY_IPC_STATUS_CONNECTED - if IPC is connected.
▶ DOCA_TELEMETRY_IPC_STATUS_DISABLED - if IPC is disabled from config.
**doca_error_t** **doca_telemetry_field_create** *(doca_telemetry_field **field)*

Create new telemetry field.

**Parameters**

- **field**
  - Pointer to the newly allocated field.

**Returns**

- **DOCA_SUCCESS** - in case of success. Error code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - NULL parameter.
  - **DOCA_ERROR_NO_MEMORY** - failed to allocate doca telemetry field.

**doca_error_t** **doca_telemetry_field_destroy** *(doca_telemetry_field *field)*

Destroy field previously created by **doca_telemetry_field_create()**.

**Parameters**

- **field**
  - Pointer to the field.

**Returns**

- **DOCA_SUCCESS** - in case of success. Error code - in case of failure:
  - **DOCA_ERROR_INVALID_VALUE** - NULL parameter.

**__DOCA_EXPERIMENTAL** **void**

**doca_telemetry_field_set_array_length** *(doca_telemetry_field *field_info, uint16_t len)*

Set doca telemetry field length.

**Parameters**

- **field_info**
  - Pointer to doca telemetry field.
- **len**
  - Field length.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

- DOCA_ERROR_INVALID_VALUE - NULL parameter(s) or invalid length (zero).

Description

Note:
If using single-value type (i.e. char) this should be 1.

__DOCA_EXPERIMENTAL void
doca_telemetry_field_set_description
(doca_telemetry_field *field_info, const char *desc)
Set doca telemetry field description.

Parameters

- field_info
  Pointer to doca telemetry field.
- desc
  Field description.

Description

Note:
Passing a field_info value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void
doca_telemetry_field_set_name
(doca_telemetry_field *field_info, const char *name)
Set doca telemetry field name.

Parameters

- field_info
  Pointer to doca telemetry field.
- name
  Field name.
Description

**Note:**
Passing a field_info value of NULL will result in an undefined behavior.

```c
__DOCA_EXPERIMENTAL void
doca_telemetry_field_set_type_name
(doca_telemetry_field *field_info, const char *type)
```

Set doca telemetry field type.

**Parameters**

- `field_info`
  Pointer to doca telemetry field.
- `type`
  Field type.

**Description**

**Note:**
Please see DOCA_TELEMETRY_FIELD_TYPE_* for possible field types

**Note:**
Passing a field_info value of NULL will result in an undefined behavior.

```c
doca_error_t
doca_telemetry_get_timestamp
(doca_telemetry_timestamp_t *timestamp)
```

Get timestamp in the proper format.

**Parameters**

- `timestamp`
  Timestamp value

**Returns**

DOCA_SUCCESS - in case of success. Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.

doca_error_t doca_telemetry_netflow_destroy (void)
Free the exporter memory and close the connection.

Returns
DOCA_SUCCESS - in case of success.

doca_error_t doca_telemetry_netflow_field_create (doca_telemetry_netflow_flowset_field **field)
Create new telemetry netflow field.

Parameters
field
Pointer to the newly allocated telemetry field.

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow field.

doca_error_t doca_telemetry_netflow_field_destroy (doca_telemetry_netflow_flowset_field *field)
Destructor for DOCA netflow field.

Parameters
field
field to destroy.

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

- DOCA_ERROR_INVALID_VALUE - if netflow_template is NULL.
DOCA_Libraries_API

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_field_set_length
(doca_telemetry_netflow_flowset_field *field,
uint16_t length)
Set doca telemetry netflow field length.

Parameters

field
  Pointer to doca telemetry netflow field.

length
  Field type.

Description

Note:
Passing a field value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_field_set_type
(doca_telemetry_netflow_flowset_field *field,
uint16_t type)
Set doca telemetry netflow field type.

Parameters

field
  Pointer to doca telemetry netflow field.

type
  Field type.

Description

Note:
Passing a field value of NULL will result in an undefined behavior.
doxa_error_t doca_telemetry_netflow_flush (void)
Immediately flush the data of the DOCA internal Netflow source.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
▷ DOCA_ERROR_BAD_STATE - if the netflow has not been started.

doca_error_t
doca_telemetry_netflow_get_buffer_data_root (const char **path)
Get data root path.

Parameters
path
The buffer data root

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
▷ DOCA_ERROR_INVALID_VALUE - NULL parameter.
▷ DOCA_ERROR_NO_MEMORY - failed to allocate memory.

doca_error_t
doca_telemetry_netflow_get_buffer_size (uint64_t *size)
Get buffer size.

Parameters
size
The buffer size

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
▷ DOCA_ERROR_INVALID_VALUE - NULL parameter.
doca_error_t
doca_telemetry_netflow_get_file_write_max_age
(doca_telemetry_timestamp_t *max_age)
Get file maximum age.

Parameters

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

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t
doca_telemetry_netflow_get_file_write_max_size
(size_t *size)
Get file maximum size.

Parameters

size
Maximum size of binary data file.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t
doca_telemetry_netflow_get_ipc_sockets_dir (const char **path)
Get IPC socket directory.

Parameters

path
Path to a folder containing DOCA Telemetry Service (DTS) sockets.
Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:
Ownership of the returned string is transferred to the caller.

doca_error_t doca_telemetry_netflow_init (uint16_t source_id)

Init exporter memory, set configs and open connection.

Parameters

source_id
Unique source ID.

Returns

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

- DOCA_ERROR_BAD_STATE - if the netflow has been initialized before this call.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.
- DOCA_ERROR_INITIALIZATION - failed to initialise netflow.

Description

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

This function can be called again only after doca_telemetry_netflow_destroy was called.
doca_error_t doca_telemetry_netflow_send (const
doca_telemetry_netflow_template *netflow_template,
const void **records, size_t nof_records, size_t
*nof_records_sent)

Sending netflow records. Need to init first.

Parameters

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

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

nof_records
Records array size.

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

Returns

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

► DOCA_ERROR_BAD_STATE - if the netflow has not been initialized or the netflow has
started.
► DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:
When sending more than 30 records the lib splits the records to multiple packets because each
packet can only send up to 30 records [Netflow protocol limit]
__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_buffer_data_root (const char *path)
Set buffer data root Default path is "/opt/mellanox/doca/services/telemetry/data/".

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

Description

Note:
This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_buffer_size (uint64_t size)
Set buffer size Default value is 60000 bytes.

Parameters
size
Buffer size

Description

Note:
This function should be called after doca_telemetry_netflow_init().
__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_collector_addr (const char *collector_addr)
Set collector address.

Parameters

collector_addr
User defined netflow collector’s IP address.

Description

Note:
This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_collector_port (uint16_t collector_port)
Set collector port. See DOCA_NETFLOW_DEFAULT_PORT for default value.

Parameters

collector_port
User defined netflow collector’s port.

Description

Note:
This function should be called after doca_telemetry_netflow_init().
__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_file_write_enabled (void)
Enable file write file write is disabled by default.

Description

Note:
This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_file_write_max_age
(doca_telemetry_timestamp_t max_age)
Set file maximum age Default value is 1 hour.

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

Description

Note:
This function should be called after doca_telemetry_netflow_init().

__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_file_write_max_size
(size_t size)
Set file maximum size Default value is 1MB.

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

**Note:**
This function should be called after `doca_telemetry_netflow_init()`.

```c
__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_ipc_enabled (void)
```

Enable IPC. IPC is disabled by default.

### Description

**Note:**
This function should be called after `doca_telemetry_netflow_init()`.

```c
__DOCA_EXPERIMENTAL void
doca_telemetry_netflow_set_ipc_sockets_dir (const char *path)
```

Set IPC socket directory. Default path is "/opt/mellanox/doca/services/telemetry/ipcsockets".

#### Parameters

- **path**
  Path to a folder containing DOCA Telemetry Service (DTS) sockets.

### Description

**Note:**
This function should be called after `doca_telemetry_netflow_init()`.
**__DOCA_EXPERIMENTAL void doca_telemetry_netflow_set_max_packet_size (uint16_t max_packet_size)**

Set max packet size.

**Parameters**

**max_packet_size**
- User defined netflow packet’s max size.

**Description**

**Note:**
This function should be called after `doca_telemetry_netflow_init()`.

**__DOCA_EXPERIMENTAL void doca_telemetry_netflow_source_set_id (const char *source_id)**

Set source id.

**Parameters**

**source_id**
- Hostname or guid.

**Description**

**Note:**
This function should be called after `doca_telemetry_netflow_init()`.
__DOCA EXPERIMENTAL void
doca_telemetry_netflow_source_set_tag (const char *source_tag)
Set source tag.

Parameters

**source_tag**
User defined data-file name prefix.

Description

### Note:
This function should be called after doca_telemetry_netflow_init().

doca_error_t doca_telemetry_netflow_start (void)
Finalizes netflow setup.

Returns

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

- **DOCA_ERROR_BAD_STATE** - if the netflow has not been initialized or the netflow has started.
- **DOCA_ERROR_NO_MEMORY** - failed to allocate memory.

doca_error_t
doca_telemetry_netflow_template_add_field
doca_telemetry_netflow_template
*netflow_template,
doca_telemetry_netflow_flowset_field *field)
Add DOCA telemetry netflow field to netflow_template. The user loses the ownership of the field after a successful invocation of the function.

Parameters

**netflow_template**
Pointer to netflow_template.
field

DOCA Telemetry netflow field to add.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow field.

Description

Note:

field should NOT be passed to another group after calling this function.

doca_error_t
doca_telemetry_netflow_template_create
doca_telemetry_netflow_template

Create new telemetry netflow template.

Parameters

netflow_template

Pointer to the newly allocated telemetry netflow template.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry netflow template.
`doca_error_t doca_telemetry_netflow_template_destroy (doca_telemetry_netflow_template *netflow_template)`

Destructor for DOCA netflow template.

**Parameters**

- `netflow_template`  
  netflow template to destroy.

**Returns**

- `DOCA_SUCCESS` - in case of success. Error code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - if `netflow_template` is NULL.

`doca_error_t doca_telemetry_schema_add_type (doca_telemetry_schema *doca_schema, const char *new_type_name, doca_telemetry_type *type, doca_telemetry_type_index_t *type_index)`

Add user-defined fields to create new type in DOCA schema. The users loses the ownership of the type after a successful invocation of the function.

**Parameters**

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

**Returns**

- `DOCA_SUCCESS` - in case of success. Error code - in case of failure:
  - `DOCA_ERROR_NO_MEMORY` - in case of memory allocation failure.
  - `DOCA_ERROR_INVALID_VALUE` - If type name exists or any of the fields have invalid field type.
**doca_error_t doca_telemetry_schema_destroy**
*(doca_telemetry_schema *doca_schema)*
Destructor for DOCA schema.

**Parameters**
- **doca_schema**
  Schema to destroy.

**Returns**
DOCA_SUCCESS - in case of success. Error code - in case of failure:
- **DOCA_ERROR_INVALID_VALUE** - if doca_schema is NULL.

**doca_error_t doca_telemetry_schema_get_buffer_data_root**
*(doca_telemetry_schema *doca_schema, const char **path)*
Get data root path.

**Parameters**
- **doca_schema**
  Pointer to DOCA schema.
- **path**
  Path to a folder where the data and schema will be stored.

**Returns**
DOCA_SUCCESS - in case of success. Error code - in case of failure:
- **DOCA_ERROR_INVALID_VALUE** - NULL parameter.
- **DOCA_ERROR_NO_MEMORY** - failed to allocate memory.

**Description**

**Note:**
Ownership of the returned string is transferred to the caller.
doca_error_t
doca_telemetry_schema_get_buffer_size
doca_telemetry_schema *doca_schema, uint64_t *size)
Get buffer size.

Parameters

doca_schema
   Pointer to DOCA schema.

size
   The buffer size

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
   DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t
doca_telemetry_schema_get_file_write_max_age
doca_telemetry_schema *doca_schema,
doca_telemetry_timestamp_t *max_age)
Get file maximum age.

Parameters

doca_schema
   Pointer to DOCA schema.

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

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
   DOCA_ERROR_INVALID_VALUE - NULL parameter.
doca_error_t
doca_telemetry_schema_get_file_write_max_size
(doca_telemetry_schema *doca_schema, size_t *size)
Get file maximum size.

Parameters

doca_schema
    Pointer to DOCA schema.
size
    Maximum size of binary data file.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - NULL parameter.

doca_error_t
doca_telemetry_schema_get_ipc_reconnect_time
(doca_telemetry_schema *doca_schema, uint32_t
*max_time)
Get IPC reconnect time in milliseconds.

Parameters

doca_schema
    Pointer to DOCA schema.
max_time
    Maximum reconnect time in milliseconds

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - NULL parameter.
**doca_error_t**  
**doca_telemetry_schema_get_ipc_reconnect_tries**(doca_telemetry_schema *doca_schema, uint8_t *tries)  
Get maximum IPC reconnect tries.  

Parameters  
**doca_schema**  
Pointer to DOCA schema.  
**tries**  
Maximum reconnect tries

Returns  
DOCA_SUCCESS - in case of success. Error code - in case of failure:  
- DOCA_ERROR_INVALID_VALUE - NULL parameter.

**doca_error_t**  
**doca_telemetry_schema_get_ipc_socket_timeout**(doca_telemetry_schema *doca_schema, uint32_t *timeout)  
Get IPC socket timeout in milliseconds.  

Parameters  
**doca_schema**  
Pointer to ipc timeout attribute.  
**timeout**  
Maximum socket timeout in milliseconds

Returns  
DOCA_SUCCESS - in case of success. Error code - in case of failure:  
- DOCA_ERROR_INVALID_VALUE - NULL parameter.
doca_telemetry_schema_get_ipc_sockets_dir

(docoa_telemetry_schema *doca_schema, const char **sockets_dir)

Get IPC socket directory.

Parameters

doca_schema
  Pointer to DOCA schema.

sockets_dir
  Path to a folder containing DOCA Telemetry Service (DTS) sockets.

Returns

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate memory.

Description

Note:
Ownership of the returned string is transferred to the caller.

doca_telemetry_schema_init

(docoa_telemetry_schema *doca_schema)

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

Parameters

schema_name
  Name of the schema.

doca_schema
  Pointer to DOCA schema, NULL on error.
Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:

- DOCA_ERROR_NO_MEMORY - failed to allocate doca_schema.
- DOCA_ERROR_INITIALIZATION - failed to initialise doca_schema.
- DOCA_ERROR_INVALID_VALUE - invalid input/output parameters.

__DOCA EXPERIMENTAL void
doca_telemetry_schema_set_buffer_data_root
(doca_telemetry_schema *doca_schema, const char *path)
Set buffer data root Default path is "/opt/mellanox/doca/services/telemetry/data/".

Parameters
doca_schema
  Pointer to DOCA schema.
path
  Path to a folder where the data and schema will be stored.

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA EXPERIMENTAL void
doca_telemetry_schema_set_buffer_size
(doca_telemetry_schema *doca_schema, uint64_t size)
Set buffer size Default value is 60000 bytes.

Parameters
doca_schema
  Pointer to DOCA schema.
size
  Buffer size
__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_file_write_enabled
(doca_telemetry_schema *doca_schema)
Enable file write file write is disabled by default.

Parameters
doca_schema
  Pointer to DOCA schema.

__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_file_write_max_age
(doca_telemetry_schema *doca_schema,
doca_telemetry_timestamp_t max_age)
Set file maximum age Default value is 1 hour.

Parameters
doca_schema
  Pointer to DOCA schema.
max_age
  Maximum file age. Once current file is older than this threshold a new file will be created.
__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_file_write_max_size
(doca_telemetry_schema *doca_schema, size_t size)
Set file maximum size Default value is 1MB.

Parameters

doca_schema
    Pointer to DOCA schema.

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

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_ipc_enabled
(doca_telemetry_schema *doca_schema)
Enable IPC IPC is disabled by default.

Parameters

doca_schema
    Pointer to DOCA schema.

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.
__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_ipc_reconnect_time
(doca_telemetry_schema *doca_schema, uint32_t max_time)
Set IPC reconnect time in milliseconds. Time limit for reconnect attempts. If the limit is reached, the client is considered disconnected. Default value is 100 milliseconds.

Parameters

doca_schema
   Pointer to DOCA schema.
max_time
   Maximum reconnect time in milliseconds

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_ipc_reconnect_tries
(doca_telemetry_schema *doca_schema, uint8_t tries)
Set maximum IPC reconnect tries. Number of reconnect attempts during reconnection period. Default value is 3 tries.

Parameters

doca_schema
   Pointer to DOCA schema.
tries
   Maximum reconnect tries

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.
__DOCA.EXPERIMENTAL void
doca-telemetry-schema-set-ipc-socket-timeout
(doca-telemetry-schema *doca_schema, uint32_t timeout)

Set IPC socket timeout in milliseconds. Timeout for IPC messaging socket. If timeout is reached during send/receive, the client is considered disconnected. Default value is 3000 milliseconds.

Parameters

- **doca_schema**: Pointer to ipc timeout attribute.
- **timeout**: Maximum socket timeout in milliseconds

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.

__DOCA.EXPERIMENTAL void
doca-telemetry-schema-set-ipc-sockets-dir
(doca-telemetry-schema *doca_schema, const char *sockets_dir)

Set IPC socket directory. Default path is "/opt/mellanox/doca/services/telemetry/ipc_sockets".

Parameters

- **doca_schema**: Pointer to DOCA schema.
- **sockets_dir**: Path to a folder containing DOCA Telemetry Service (DTS) sockets

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.
__DOCA_EXPERIMENTAL void
doca_telemetry_schema_set_opaque_events_enabled
(doca_telemetry_schema *doca_schema)
Enable opaque events Opaque events are disabled by default.

Parameters

doca_schema
Pointer to DOCA schema.

Description

Note:
Passing a doca_schema value of NULL will result in an undefined behavior.

doca_error_t doca_telemetry_schema_start
(doca_telemetry_schema *doca_schema)
Finalizes schema setup to start creating Doca Sources from the schema.

Parameters

doca_schema
Input schema to start.

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

- DOCA_ERROR_INITIALIZATION - in case of failure.

Description
Do NOT add new types after this function was called.
docta_error_t docta_telemetry_source_create
(doca_telemetry_schema *doca_schema,
doca_telemetry_source **doca_source)
Creates a single DOCA source from schema.

Parameters

doca_schema
  Schema from which source will be created.
doca_source
  pointer to DOCA source, or NULL on error.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

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

docta_error_t docta_telemetry_source_destroy
(doca_telemetry_source *doca_source)
Destructor for DOCA source.

Parameters

doca_source
  Source to destroy.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  ▶ DOCA_ERROR_INVALID_VALUE - if doca_source is NULL.
`doca_error_t doca_telemetry_source_flush`  
(doca_telemetry_source *doca_source)

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

**Parameters**

`doca_source`  
DOCA source to flush.

**Returns**

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

- `DOCA_ERROR_INVALID_VALUE` - if `doca_source` is NULL.

`doca_error_t`  
`doca_telemetry_source_get_opaque_report_max_data_size`  
(doca_telemetry_source *doca_source, uint32_t *max_data_size)

Get max data size for opaque report.

**Parameters**

`doca_source`  
Source to report.

`max_data_size`  
Maximal data size

**Returns**

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

- `DOCA_ERROR_INVALID_VALUE` - NULL parameter(s).
Report opaque event data via DOCA source.

**Parameters**

- `doca_source`  
  Source to report.
- `app_id`  
  User defined application ID.
- `user_defined1`  
  User defined parameter 1.
- `user_defined2`  
  User defined parameter 2.
- `data`  
  Data buffer.
- `data_size`  
  Size of the data in the data buffer.

**Returns**

- DOCA_SUCCESS - in case of success. Error code - in case of failure:
  - DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

**Description**

Data is flushed from internal buffer when the buffer is full. Flushing the data immediately can be done by invoking `doxa_telemetry_source_flush()`.
doса_error_t doca_telemetry_source_report
doca_telemetry_source *doca_source,
doca_telemetry_type_index_t index, void *data, int
count)

Report events data of the same type via DOCA source.

Parameters

doса_source
  Source to report.

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

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
  • DOCA_ERROR_NO_MEMORY - in case of memory allocation failure.

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

__DOCA_EXPERIMENTAL void
doca_telemetry_source_set_id
(doca_telemetry_source *doca_source, const char
*source_id)

Set source id.

Parameters

doса_source
  Pointer to DOCA source.

source_id
  Hostname or guid.
## Modules

### DOCA Libraries API

### v283

---

**Description**

**Note:**
Passing a `doca_source` value of NULL will result in an undefined behavior.

```c
.DOCA_EXPERIMENTAL void
doca_telemetry_source_set_tag
(doca_telemetry_source *doca_source, const char *source_tag)
```

Set source tag.

**Parameters**

- `doca_source`
  - Pointer to DOCA source.
- `source_tag`
  - User defined data-file name prefix.

### doca_error_t doca_telemetry_source_start

```c
(doca_telemetry_source *doca_source)
```

Applies source attribute and starts DOCA source.

**Parameters**

- `doca_source`
  - DOCA source to start.

**Returns**

- `DOCA_SUCCESS` - in case of success. Error code - in case of failure:
  - `DOCA_ERROR_INVALID_VALUE` - if source attributes are not set.
  - `DOCA_ERROR_NO_MEMORY` - in case of memory allocation failure.
Description
Call this function to start reporting.

doca_error_t doca_telemetry_type_add_field (doca_telemetry_type *type, doca_telemetry_field *field)
Add DOCA telemetry field to type. The users loses the ownership of the field after a successful invocation of the function.

Parameters
- type
  Pointer to doca telemetry type.
- field
  DOCA Telemetry field to add.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
- DOCA_ERROR_INVALID_VALUE - NULL parameter.
- DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry field.

Note:
field should NOT be passed to another type after calling this function.

doca_error_t doca_telemetry_type_create (doca_telemetry_type **type)
Create new telemetry type.

Parameters
- type
  Pointer to the newly allocated type.

Returns
DOCA_SUCCESS - in case of success. Error code - in case of failure:
DOCA_ERROR_INVALID_VALUE - NULL parameter.
DOCA_ERROR_NO_MEMORY - failed to allocate doca telemetry field.

**doca_error_t doca_telemetry_type_destroy**(doca_telemetry_type *type)
Destroy doca telemetry type previously created by doca_telemetry_type_create().

**Parameters**

**type**
Pointer to type.

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

- DOCA_ERROR_INVALID_VALUE - NULL parameter.

**Description**

```
#define DOCA_GUID_SIZE 16
DOCA GUID size.

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

```
#define DOCA_NETFLOW_DEFAULT_PORT 2055
NetFlow collector default port.
```

**Note:**
fields added to this type should NOT be used after calling this function.

**Note:**
This GUID cannot change.
#define DOCA_TELEMETRY_FIELD_TYPE_BOOL "bool"
DOCA_TELEMETRY_FIELD_TYPE_{} are data types that are used to create
doca_telemetry_field,

DOCA telemetry bool type

#define DOCA_TELEMETRY_FIELD_TYPE_CHAR "char"
DOCA telemetry char type.

#define DOCA_TELEMETRY_FIELD_TYPE_DOUBLE "double"
DOCA telemetry double type.

#define DOCA_TELEMETRY_FIELD_TYPE_FLOAT "float"
DOCA telemetry float type.

#define DOCA_TELEMETRY_FIELD_TYPE_IN "int"
DOCA telemetry int type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT16 "int16_t"
DOCA telemetry int16 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT32 "int32_t"
DOCA telemetry int32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_INT64 "int64_t"
DOCA telemetry int64 type.
#define DOCA_TELEMETRY_FIELD_TYPE_INT8
"int8_t"
DOCA telemetry int8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONG
"long"
DOCA telemetry long type.

#define DOCA_TELEMETRY_FIELD_TYPE_LONGLONG
"long long"
DOCA telemetry longlong type.

#define DOCA_TELEMETRY_FIELD_TYPE_SHORT
"short"
DOCA telemetry short type.

#define DOCA_TELEMETRY_FIELD_TYPE_TIMESTAMP
DOCA telemetry timestamp type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT64
DOCA telemetry uint64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UCHAR
"unsigned char"
DOCA telemetry uchar type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT
"unsigned int"
DOCA telemetry uint type.

#define DOCA_TELEMETRY_FIELD_TYPE UINT16
"uint16_t"
DOCA telemetry uint16 type.
#define DOCA_TELEMETRY_FIELD_TYPE_UINT32 "uint32_t"
DOCA telemetry uint32 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT64 "uint64_t"
DOCA telemetry uint64 type.

#define DOCA_TELEMETRY_FIELD_TYPE_UINT8 "uint8_t"
DOCA telemetry uint8 type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONG "unsigned long"
DOCA telemetry ulong type.

#define DOCA_TELEMETRY_FIELD_TYPE_ULONGLONG "long long"
DOCA telemetry ulonglong type.

#define DOCA_TELEMETRY_FIELD_TYPE_USHORT "unsigned short"
DOCA telemetry ushort type.

2.23. Version Management

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

const char *doca_version (void)
Function returning DOCA’s [SDK] version string.

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

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

```c
const __DOCA_EXPERIMENTAL char *
doca_version_runtime (void)
```
Function returning DOCA’s (runtime) version string.

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

**Description**

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

```c
#define DOCA_CURRENT_VERSION_NUM
DOCA_VERSION_NUM(DOCA_VER_MAJOR,
DOCA_VER_MINOR, DOCA_VER_PATCH)
```
Macro of current version number for comparisons.

```c
#define DOCA_VER_MAJOR 1
```
Major version number 0-255.

```c
#define DOCA_VER_MINOR 5
```
Minor version number 0-255.

```c
#define DOCA_VER_PATCH 1007
```
Patch version number 0-9999.

```c
#define DOCA_VER_STRING "1.5.1007"
```
DOCA Version String.
#define DOCA_VERSION_EQ_CURRENT 
(DOCA_VERSION_NUM(major, minor, patch) == 
DOCA_CURRENT_VERSION_NUM)

Check if the version specified is equal to current.

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

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

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

Macro of version number for comparisons.

2.3. Change Log

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

1.3.0

- Field Groups, GPU Groups, and field watches created with a handle returned from dcgmConnect() are now cleaned up upon disconnect. dcgmConnect_v2() can be used to get the old behavior of objects persisting after disconnect.
- dcgmConnect_v2() was added as a method for specifying additional connection options when connecting to the host engine.
- dcgmUnwatchFields() was added as a method of unwatching fields that were previously watched with dcgmWatchFields()
- dcgmActionValidate_v2() was added to be able to pass more parameters to the DCGM GPU Diagnostic.
- dcgmDiagResponse_t was increased from v2 to v3. See dcgmDiagResponse_v3 for details

1.2.3

- No API changes in this version.
1.1.1

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

1.0.0

- Initial Release.
Chapter 3. Data Structures

Here are the data structures with brief descriptions:

- **doca_compress_job**: 
- **doca_ct_cfg**: Doca ct global configuration
- **doca_dma_job_memcpy**: 
- **doca_dma_memcpy_result**: 
- **doca_dpi_config_t**: DPI init configuration
- **doca_dpi_grpc_generic_packet**: Generic packet that holds payload or a whole packet as segment
- **doca_dpi_grpc_result**: Dequeue result
- **doca_dpi_parsing_info**: L2-L4 flow information
- **doca_dpi_result**: Dequeue result
- **doca_dpi_sig_data**: Extra signature data
- **doca_dpi_sig_info**: Signature info
- **doca_dpi_stat_info**: DPI statistics
- **doca_encryption_key**: IPSec encryption key
- **doca_event**: Activity completion event
- **doca_flow_action_desc**: Action description
- **doca_flow_action_descs**: Action descriptions
- **doca_flow_action_descs_meta**: Metadata action description per field
**doca_flow_action_field**  
Extended modification action

**doca_flow_actions**  
Doca flow actions information

**doca_flow_aged_query**  
Aged flow query callback context

**doca_flow_cfg**  
Doca flow global configuration

**doca_flow_encap_action**  
Doca flow encap data information

**doca_flow_error**  
Doca flow error message struct

**doca_flow_fwd**  
Forwarding configuration

**doca_flow_grpcBindable_obj**  
Bindable object configuration

**doca_flow_grpc_fwd**  
Forwarding configuration wrapper

**doca_flow_grpc_pipe_cfg**  
Pipeline configuration wrapper

**doca_flow_grpc_response**  
General DOCA Flow response struct

**doca_flow_ip_addr**  
Doca flow ip address

**doca_flow_match**  
Doca flow matcher information

**doca_flow_meta**  
Doca flow meta data

**doca_flow_monitor**  
Doca monitor action configuration

**doca_flow_ordered_list**

**doca_flow_pipe_attr**  
Pipe attributes

**doca_flow_pipe_cfg**  
Pipeline configuration

**doca_flow_port_cfg**  
Doca flow port configuration

**doca_flow_query**  
Flow query result

**doca_flow_resource_crypto_cfg**  
Doca flow crypto resource configuration

**doca_flow_resource_meter_cfg**  
Doca flow meter resource configuration
**doca_flow_resource_rss_cfg**
Doca flow rss resource configuration

**doca_flow_resources**
Doca flow resource quota

**doca_flow_shared_resource_cfg**
Doca flow shared resource configuration

**doca_flow_shared_resource_result**
Flow shared resources query result

**doca_flow_tun**
Doca flow tunnel information

**doca_ipsec_sa_antireplay**
IPSec antireplay attributes, part of ipsec attr

**doca_ipsec_sa_attrs**
IPSec attributes to create jobs

**doca_ipsec_sa_create_job**
DOCA IPSec SA creation job

**doca_ipsec_sa_destroy_job**
DOCA IPSec SA destroy job

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

**doca_log_registrator**
Registers log source on program start

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

**doca_regex_job_search**

**doca_regex_match**

**doca_regex_search_result**

**doca_rmax_cpu_affinity_mask**
Data structure to describe CPU mask for doca_rmax internal thread

**doca_rmax_in_stream_completion**
Completion returned by input stream describing the incoming packets

**doca_rmax_stream_error**
Detailed completion error information

**doca_sha_job**

**doca_sha_partial_job**

### 3.1. **doca_compress_job** Struct Reference

Jobs to be dispatched via COMPRESS library.
struct doca_job doca_compress_job::base

Common job data.

doca_buf *doca_compress_job::dst_buff

Destination data buffer.

uint64_t *doca_compress_job::output_chksum

Output checksum. If it is a compress job the checksum calculated is of the src_buf. If it is a decompress job the checksum result calculated is of the dst_buf. When the job processing will end, the output_chksum will contain the CRC checksum result in the lower 32bit and the Adler checksum result in the upper 32bit.

const doca_buf *doca_compress_job::src_buff

Source data buffer.

3.2.  doca_ct_cfg Struct Reference

doca ct global configuration

uint32_t doca_ct_cfg::flags

CT behavior flags

void *doca_ct_cfg::ib_dev

IB verbs device context

void *doca_ct_cfg::ib_pd

device protection domain

uint32_t doca_ct_cfg::nb_queues

number of CT queues(thread).
3.3.  `doca_dma_job_memcpy` Struct Reference

A job to be dispatched via the DMA library.

```c
struct doca_job doca_dma_job_memcpy::base
```

Common job data

```c
doca_buf *doca_dma_job_memcpy::dst_buff
```

Destination data buffer

```c
const doca_buf *doca_dma_job_memcpy::src_buff
```

Source data buffer

3.4.  `doca_dma_memcpy_result` Struct Reference

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

```c
doca_error_t doca_dma_memcpy_result::result
```

Operation result

3.5.  `doca_dpi_config_t` Struct Reference

DPI init configuration.

```c
uint32_t doca_dpi_config_t::max_packets_per_queue
```

Number of packets concurrently processed by the DPI engine.

```c
uint32_t doca_dpi_config_t::max_sig_match_len
```

The maximum length that DPI guarantee to provide a match on, including across consecutive packets. Must be <= 5000 For example: Signature = A.*B max_sig_match_len = 5 DPI
guarantee that AAAAB will be found (len <= 5) DPI does not guarantee that AAAAAAAAAAAAB will be found (len > 5)

The minimum required overlap between two packets for regex match

```c
uint16_t doca_dpi_config_t::nb_queues
```

Number of DPI queues

```c
const char *doca_dpi_config_t::server_address
```

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

### 3.6. doca_dpi_grpc_generic_packet Struct Reference

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

```c
uint16_t doca_dpi_grpc_generic_packet::seg_len
```

The length of the data inside segment buffer

```c
uint8_t *doca_dpi_grpc_generic_packet::segment
```

The buffer with data to be scanned by the DPI

### 3.7. doca_dpi_grpc_result Struct Reference

Dequeue result.

```c
struct doca_dpi_sig_info doca_dpi_grpc_result::info
```

Signature information

```c
bool doca_dpi_grpc_result::matched
```

Indicates flow was matched
doca_dpi_grpc_generic_packet
*doca_dpi_grpc_result::pkt

Pkt provided on enqueue

int doca_dpi_grpc_result::status_flags
doca_dpi_flow_status flags

void *doca_dpi_grpc_result::user_data
User data provided on enqueue

3.8. doca_dpi_parsing_info Struct
Reference

L2-L4 flow information.

doca_dpi_parsing_info::@2
doca_dpi_parsing_info::dst_ip

IP destination address

doca_dpi_parsing_info::@0
doca_dpi_parsing_info::dst_ip

IP destination address

__be16 doca_dpi_parsing_info::ethertype

Ethertype of the packet in network byte order

in_addr doca_dpi_parsing_info::ipv4

Ipv4 destination address in network byte order

Ipv4 source address in network byte order
3.9. **doxa_dpi_result** Struct Reference

Deque result.

```c
struct doxa_dpi_sig_info doxa_dpi_result::info
```
Signature information

```c
bool doxa_dpi_result::matched
```
Indicates flow was matched
rte_mbuf *doca_dpi_result::pkt
Pkt provided on enqueue

int doca_dpi_result::status_flags
doca_dpi_flow_status flags

void *doca_dpi_result::user_data
User data provided on enqueue

3.10. doca_dpi_sig_data Struct Reference
Extra signature data.

char doca_dpi_sig_data::name
Signature name

uint32_t doca_dpi_sig_data::sig_id
Signature ID as provided in the signature

3.11. doca_dpi_sig_info Struct Reference
Signature info.

int doca_dpi_sig_info::action
The action as provided in the signature

uint32_t doca_dpi_sig_info::sig_id
Signature ID as provided in the signature

3.12. doca_dpi_stat_info Struct Reference
DPI statistics.
uint32_t doca_dpi_stat_info::nb_http_parser_based
Total number of http signature matches

uint32_t doca_dpi_stat_info::nb_matches
Total number of signature matches

uint32_t doca_dpi_stat_info::nb_other_l4
Total number of other l4 signature matches

uint32_t doca_dpi_stat_info::nb_other_l7
Total number of other l7 signature matches

uint32_t doca_dpi_stat_info::nb_scannedPkts
Total number of scanned packets

uint32_t doca_dpi_stat_info::nb_ssl_parser_based
Total number of ssl signature matches

uint32_t doca_dpi_stat_info::nb_tcp_based
Total number of tcp signature matches

uint32_t doca_dpi_stat_info::nb_udp_based
Total number of udp signature matches

3.13. doca_encryption_key Struct
Reference

IPSec encryption key.

uint64_t doca_encryption_key::implicit_iv
The IV is inserted into the GCM engine is calculated by
void *doca_encryption_key::raw_key
Raw key buffer. Actual size of this buffer defined by type.

uint32_t doca_encryption_key::salt
The salt is inserted into the GCM engine is calculated by

doca_encryption_key_type
doca_encryption_key::type
size of enc key

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

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

int doca_event::type
The type of the event originating activity.

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

3.15.  doca_flow_action_desc Struct Reference
action description
enum doca_flow_action_type

doca_flow_action_desc::type

type

3.16. doca_flow_action_descs Struct Reference

action descriptions

struct doca_flow_action_desc
doca_flow_action_descs::dst_ip

action description of destination IP.

struct doca_flow_action_desc
doca_flow_action_descs::dst_mac

action description of destination MAC.

struct doca_flow_action_desc
doca_flow_action_descs::dst_port

action description of destination L4 port.

struct doca_flow_action_desc
doca_flow_action_descs::eth_type

action description of ether type.

struct doca_flow_action_descs_meta
doca_flow_action_descs::meta

action description of meta data.
struct doca_flow_action_desc
doca_flow_action_descs::src_ip

action description of source IP.

struct doca_flow_action_desc
doca_flow_action_descs::src_mac

action description of source MAC.

struct doca_flow_action_desc
doca_flow_action_descs::src_port

action description of source L4 port.

struct doca_flow_action_desc
doca_flow_action_descs::ttl

action description of IPv4 TTL.

struct doca_flow_action_desc
doca_flow_action_descs::tunnel

action description of tunnel.

struct doca_flow_action_desc
doca_flow_action_descs::vlan_id

action description of VLAN ID.

3.17. doca_flow_action_descs_meta Struct
Reference

Metadata action description per field.
struct doca_flow_action_desc

doca_flow_action_descs_meta::pkt_meta

action description of pkt_meta.

struct doca_flow_action_desc

doca_flow_action_descs_meta::u32

action description of meta.

3.18. doca_flow_action_field Struct Reference

extended modification action

void *doca_flow_action_field::address

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

uint32_t doca_flow_action_field::offset

If address is not NULL, bit offset within the field from the address. Otherwise, bit offset from the start of context field.

3.19. doca_flow_actions Struct Reference

doca flow actions information

uint8_t doca_flow_actions::action_idx

index according to place provided on creation

uint32_t doca_flow_actions::crypto_id

Crypto shared action id

bool doca_flow_actions::decap

when true, will do decap
struct doca_flow_encap_action

doca_flow_actions::encap
encap data information

uint32_t doca_flow_actions::flags

action flags

bool doca_flow_actions::has_encap

when true, will do encap

struct doca_flow_meta doca_flow_actions::meta

modify meta data, pipe action as mask

struct doca_flow_ip_addr
doca_flow_actions::mod_dst_ip

modify destination ip address

uint8_t doca_flow_actions::mod_dst_mac

modify VLAN ID

doca_be16_t doca_flow_actions::mod_dst_port

modify layer 4 destination port

struct doca_flow_ip_addr
doca_flow_actions::mod_src_ip

modify source ip address

uint8_t doca_flow_actions::mod_src_mac

modify source mac address

doca_be16_t doca_flow_actions::mod_src_port

modify layer 4 source port
doca_be16_t doca_flow_actions::mod_vlan_id
modify destination mac address

doc_flow_crypto_protocol_type
doca_flow_actions::proto_type
Crypto shared action type

doca_flow_actions::@8 doca_flow_actions::security
security shared action

uint8_t doca_flow_actions::ttl
modify[ADD] TTL value

3.20. doca_flow_aged_query Struct Reference
aged flow query callback context

uint64_t doca_flow_aged_query::user_data
The user input context, otherwise the doca_flow_pipe_entry pointer

3.21. doca_flow_cfg Struct Reference
doca flow global configuration

doca_flow_entry_process_cb doca_flow_cfg::cb
callback for entry create/destroy

const char *doca_flow_cfg::mode_args
set doca flow architecture mode switch, vnf

uint32_t doca_flow_cfg::nr_shared_resources
total shared resource per type
uint32_t doca_flow_cfg::queue_depth
Number of pre-configured queue_size, default to 128

uint16_t doca_flow_cfg::queues
queue id for each offload thread

struct doca_flow_resources doca_flow_cfg::resource
resource quota

doca_flow_shared_resource_unbind_cb
doca_flow_cfg::unbind_cb
callback for unbinding of a shared resource

3.22. doca_flow_encap_action Struct
Reference
doca flow encap data information

struct doca_flow_ip_addr
doca_flow_encap_action::dst_ip
destination ip address

uint8_t doca_flow_encap_action::dst_mac
destination mac address

struct doca_flow_ip_addr
doca_flow_encap_action::src_ip
source ip address

uint8_t doca_flow_encap_action::src_mac
source mac address
struct doca_flow_tun doca_flow_encap_action::tun
  tunnel info
doca_be16_t doca_flow_encap_action::vlan_tci
  vlan tci

3.23. doca_flow_error Struct Reference
doca flow error message struct
const char *doca_flow_error::message
  Human-readable error message
enum doca_flow_error_type doca_flow_error::type
  Cause field and error types

3.24. doca_flow_fwd Struct Reference
  forwarding configuration
uint32_t doca_flow_fwd::idx
  Index of the ordered list pipe entry.
doca_flow_pipe *doca_flow_fwd::next_pipe
  next pipe pointer
int doca_flow_fwd::num_of_queues
  number of queues
doca_flow_fwd::@9::@17
doca_flow_fwd::ordered_list_pipe
  next ordered list pipe configuration
doce_flow_pipe *doca_flow_fwd::pipe
Ordered list pipe to select an entry from.

uint16_t doca_flow_fwd::port_id
destination port id

uint32_t doca_flow_fwd::rss_flags
rss offload types

uint16_t *doca_flow_fwd::rss_queues
rss queues array

uint32_t doca_flow_fwd::shared_rss_id
shared rss id, only for pipe’s fwd is NULL

enum doca_flow_fwd_type doca_flow_fwd::type
indicate the forwarding type

3.25. doca_flow_grpc_bindable_obj Struct

Reference

bindable object configuration

uint64_t doca_flow_grpc_bindable_obj::pipe_id
pipe id if type is pipe

uint32_t doca_flow_grpc_bindable_obj::port_id
port id if type is port

enum doca_flow_grpc_bindable_obj_type
doca_flow_grpc_bindable_obj::type
bindable object type
3.26. **doca_flow_grpc_fwd Struct Reference**

Forwarding configuration wrapper

```c
doca_flow_fwd *doca_flow_grpc_fwd::fwd
```

doca flow fwd struct

```c
uint64_t doca_flow_grpc_fwd::next_pipe_id
```

next pipe id

3.27. **doca_flow_grpc_pipe_cfg Struct Reference**

Pipeline configuration wrapper

```c
doca_flow_pipe_cfg *doca_flow_grpc_pipe_cfg::cfg
```

doca_flow_pipe_cfg struct

```c
uint16_t doca_flow_grpc_pipe_cfg::port_id
```

port id

3.28. **doca_flow_grpc_response Struct Reference**

General DOCA Flow response struct.

```c
int doca_flow_grpc_response::aging_res
```

Return value from handle aging

```c
uint64_t doca_flow_grpc_response::entry_id
```

entry id
enum doca_flow_entry_status
doca_flow_grpc_response::entry_status
return value of entry get status

struct doca_flow_error
doca_flow_grpc_response::error
Otherwise, this field contains the error information

uint64_t
doca_flow_grpc_response::nb_entries_processed
return value from entries process

uint64_t doca_flow_grpc_response::pipe_id
pipe id

bool doca_flow_grpc_response::success
in case of success should be true

uint64_t doca_flow_grpc_response::switch_port_id
switch port id

3.29. doca_flow_ip_addr Struct Reference
doca flow ip address
doca_be32_t doca_flow_ip_addr::ipv4_addr
ipv4 address if type is ipv4
doca_be32_t doca_flow_ip_addr::ipv6_addr
ipv6 address if type is ipv6
uint8_t doca_flow_ip_addr::type
ip address type

3.30. doca_flow_match Struct Reference
doca flow matcher information

uint32_t doca_flow_match::flags
match items which are no value

struct doca_flow_ip_addr doca_flow_match::in_dst_ip
inner destination ip address if tunnel is used

uint8_t doca_flow_match::in_dst_mac
inner destination mac address

doca_be16_t doca_flow_match::in_dst_port
inner layer 4 destination port if tunnel is used

doca_be16_t doca_flow_match::in_eth_type
inner Ethernet layer type

uint8_t doca_flow_match::in_l4_type
inner layer 4 protocol type if tunnel is used

struct doca_flow_ip_addr doca_flow_match::in_src_ip
inner source ip address if tunnel is used

uint8_t doca_flow_match::in_src_mac
inner source mac address

doca_be16_t doca_flow_match::in_src_port
inner layer 4 source port if tunnel is used
uint8_t doca_flow_match::in_tcp_flags
inner tcp flags

doca_be16_t doca_flow_match::in_vlan_tci
inner vlan tci

struct doca_flow_meta doca_flow_match::meta
Programmable meta data.

struct doca_flow_ip_addr
doca_flow_match::out_dst_ip
outer destination ip address

uint8_t doca_flow_match::out_dst_mac
outer destination mac address

doca_be16_t doca_flow_match::out_dst_port
outer layer 4 destination port

doca_be16_t doca_flow_match::out_eth_type
outer Ethernet layer type

uint8_t doca_flow_match::out_l4_type
outer layer 4 protocol type

struct doca_flow_ip_addr
doca_flow_match::out_src_ip
outer source ip address

uint8_t doca_flow_match::out_src_mac
outer source mac address
3.31. doca_flow_meta Struct Reference

doca flow meta data

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

uint8_t doca_flow_meta::align

Structure alignment.

uint8_t doca_flow_meta::ipsec_syndrome

IPsec decrypt/authentication syndrome.

uint32_t doca_flow_meta::lag_port

Bits of LAG member port.

uint32_t doca_flow_meta::mark

Mark id.
**3.32.  doca_flow_monitor Struct Reference**

doca monitor action configuration

uint32_t doca_flow_monitor::aging

aging time in seconds.

uint64_t doca_flow_monitor::cbs

Committed Burst Size (bytes).

uint64_t doca_flow_monitor::cir

Committed Information Rate (bytes/second).

uint8_t doca_flow_monitor::flags

indicate which actions be included
uint32_t doca_flow_monitor::shared_counter_id
shared counter id

uint32_t doca_flow_monitor::shared_meter_id
shared meter id

uint64_t doca_flow_monitor::user_data
aging user data input.

3.33. doca_flow_ordered_list Struct
Reference
Ordered list configuration.

const **doca_flow_ordered_list::elements
An array of DOCA flow structure pointers, depending on types.

uint32_t doca_flow_ordered_list::idx
List index among the lists of the pipe. At pipe creation, it must match the list position in the array of lists. At entry insertion, it determines which list to use.

uint32_t doca_flow_ordered_list::size
Number of elements in the list.

3.34. doca_flow_pipe_attr Struct
Reference
pipe attributes

bool doca_flow_pipe_attr::is_root
pipeline is root or not. If true it means the pipe is a root pipe executed on packet arrival.
const char *doca_flow_pipe_attr::name
name for the pipeline

uint8_t doca_flow_pipe_attr::nb_actions
maximum number of doca flow action array, default is 1 if not set

uint32_t doca_flow_pipe_attr::nb_flows
maximum number of flow rules, default is 8k if not set

uint8_t doca_flow_pipe_attr::nb_ordered_lists
number of ordered lists in the array, default 0, mutually exclusive with nb_actions

enum doca_flow_pipe_type doca_flow_pipe_attr::type
type of pipe. enum doca_flow_pipe_type

3.35. doca_flow_pipe_cfg Struct Reference
pipeline configuration

**doca_flow_pipe_cfg::action_descs
action array descriptions

**doca_flow_pipe_cfg::actions
actions array for the pipeline

struct doca_flow_pipe_attr doca_flow_pipe_cfg::attr
attributes of pipe

doca_flow_match *doca_flow_pipe_cfg::match
matcher for the pipeline

doca_flow_match *doca_flow_pipe_cfg::match_mask
match mask for the pipeline
doça_flow_monitor *doça_flow_pipe_cfg::monitor
monitor for the pipeline

**doça_flow_pipe_cfg::ordered_lists
array of ordered list types

doca_flow_port *doça_flow_pipe_cfg::port
port for the pipeline

3.36.  doca_flow_port_cfg Struct Reference
doca flow port configuration

const char *doça_flow_port_cfg::devargs
specific per port type cfg

uint16_t doça_flow_port_cfg::port_id
dpdk port id

uint16_t doça_flow_port_cfg::priv_data_size
user private data

enum doca_flow_port_type doça_flow_port_cfg::type
mapping type of port

3.37.  doca_flow_query Struct Reference
flow query result

uint64_t doça_flow_query::total_bytes
total bytes hit this flow
**3.38. doca_flow_resource_crypto_cfg Struct Reference**

doca flow crypro resource configuration

**enum doca_flow_crypto_action_type**

doca_flow_resource_crypto_cfg::action_type

crypto action

**struct doca_flow_fwd**

doca_flow_resource_crypto_cfg::fwd

Crypto action continuation

**enum doca_flow_crypto_header_type**

doca_flow_resource_crypto_cfg::header_type

packet header type

**uint8_t doca_flow_resource_crypto_cfg::key**

Crypto key buffer

**uint16_t doca_flow_resource_crypto_cfg::key_sz**

key size in bytes

**enum doca_flow_crypto_net_type**

doca_flow_resource_crypto_cfg::net_type

packet network mode type
enum doca_flow_crypto_protocol_type
doça_flow_resource_crypto_cfg::proto_type

packet reformat action

uint8_t
doça_flow_resource_crypto_cfg::reformat_data
reformat header buffer

uint16_t
doça_flow_resource_crypto_cfg::reformat_data_sz
reformat header length in bytes

enum doca_flow_crypto_reformat_type
doça_flow_resource_crypto_cfg::reformat_type

packet reformat action

void *doça_flow_resource_crypto_cfg::security_ctx
crypto object handle

3.39. doca_flow_resource_meter_cfg
Struct Reference
doca flow meter resource configuration

uint64_t doça_flow_resource_meter_cfg::cbs
Committed Burst Size (bytes).

uint64_t doça_flow_resource_meter_cfg::cir
Committed Information Rate (bytes/second).
3.40. **doca_flow_resource rss cfg Struct Reference**

*doca flow rss resource configuration*

`uint32_t doca_flow_resource_rss_cfg::flags`
- rss offload types

`int doca_flow_resource_rss_cfg::nr_queues`
- number of queues

`uint16_t *doca_flow_resource_rss_cfg::queues_array`
- rss queues array

3.41. **doca_flow_resources Struct Reference**

*doca flow resource quota*

`uint32_t doca_flow_resources::nb_counters`
- Number of counters to configure

`uint32_t doca_flow_resources::nb_meters`
- Number of traffic meters to configure

3.42. **doca_flow_shared_resource_cfg Struct Reference**

*doca flow shared resource configuration*
3.43. doca_flow_shared_resource_result Struct Reference
flow shared resources query result

3.44. doca_flow_tun Struct Reference
doca flow tunnel information

doca_be32_t doca_flow_tun::audp_hdr
Opaque audp tunnel header

doca_be32_t doca_flow_tun::esp_sn
ipsec sequence number

doca_be32_t doca_flow_tun::esp_spi
ipsec session parameter index

doca_be32_t doca_flow_tun::gre_key
gre key

doca_be32_t doca_flow_tun::gtp_teid
gtp teid

doca_be16_t doca_flow_tun::key_present
gre key is present

doca_be32_t doca_flow_tun::nisp_hdr
Opaque nisp tunnel header

doca_be16_t doca_flow_tun::protocol
next protocol
enum doca_flow_tun_type doca_flow_tun::type

tunnel type

doca_be32_t doca_flow_tun::vxlan_tun_id

vxlan vni[24] + reserved [8].

3.45. doca_ipsec_sa_antireplay Struct

Reference

IPSec antireplay attributes, part of ipsec attr.

uint32_t doca_ipsec_sa_antireplay::antireplay_enable

1 when enabled; 0 otherwise. Ingress: when enabled activates anti-replay protection window. Egress: when enabled increment IPSec SN.

uint32_t doca_ipsec_sa_antireplay::esn_overlap_event_arm

1 when armed/to arm 0 otherwise.

uint32_t doca_ipsec_sa_antireplay::hard_lifetime_arm

1 when armed/to arm 0 otherwise.

uint32_t doca_ipsec_sa_antireplay::remove_flow_enable

1 when remove flow enabled/to enable; 0 otherwise.

uint32_t doca_ipsec_sa_antireplay::remove_flow_packet_count

Packet counter, Decrements for every packet passing through the SA. Event are triggered occurs when the counter reaches soft-lifetime and hard-lifetime (0). When counter reaches hard-lifetime, all passing packets will return a relevant Syndrome.
uint32_t 
\texttt{doca_ipsec\_sa\_antireplay::remove\_flow\_soft\_lifetime} 

Soft Lifetime threshold value. When remove\_flow\_packet\_count reaches this value a soft 
lifetime event is triggered [if armed]. See remove\_flow\_packet\_count field in this struct fro 
more details.

uint8_t* \texttt{doca_ipsec\_sa\_antireplay::replay\_win\_state} 

Anti replay window state for query. Size of this array should be equal to replay win size. 
Ignored on SA create/update.

enum \texttt{doca\_ipsec\_replay\_win\_size} 
\texttt{doca\_ipsec\_sa\_antireplay::replay\_win\_sz} 

Anti replay window size to enable sequence replay attack handling. Ignored on egress & when 
antireplay\_enable field is 0.

uint32_t \texttt{doca\_ipsec\_sa\_antireplay::soft\_lifetime\_arm} 

1 when armed/to arm 0 otherwise.

\textit{3.46. doca\_ipsec\_sa\_attrs Struct} 
Reference

IPSec attributes to create jobs.

\texttt{struct doca\_ipsec\_sa\_antireplay} 
\texttt{doca\_ipsec\_sa\_attrs::antireplay} 
IPSec antireplay attr

\texttt{uint32\_t doca\_ipsec\_sa\_attrs::direction} 

ingress/decript - egress/encrypt

\texttt{uint32\_t doca\_ipsec\_sa\_attrs::esn\_enabled} 

when set esn is enabled
**uint32_t doca_ipsec_sa_attrs::esn_overlap**
new/old indication of the High sequence number MSB - when set is old

**enum doca_ipsec_icv_length**
**doca_ipsec_sa_attrs::icv_length**
Authentication Tag length

**struct doca_encryption_key doca_ipsec_sa_attrs::key**
IPSec encryption key

**enum doca_ipsec_sa_mode**
**doca_ipsec_sa_attrs::mode**
ipsec protocol mode - transport of tunnel

**enum doca_ipsec_sa_offload**
**doca_ipsec_sa_attrs::offload**
offload type - full or only crypto - only supported DOCA_IPSEC_SA_OFFLOAD_FULL;

**enum doca_ipsec_sa_protocol**
**doca_ipsec_sa_attrs::protocol**
protocol type - esp or ah - only supported DOCA_IPSEC_SA_PROTO_ESP

**uint32_t doca_ipsec_sa_attrs::spi**
SA security parameter index

### 3.47. **doca_ipsec_sa_create_job Struct**

**Reference**
DOCA IPSec SA creation job.
The result of this job if doca_workq_progress_retrieve returns:

- **DOCA_SUCCESS** - struct `doca_event {.result.ptr }` should point to new created `struct doca_ipsec_sa` object.
- DOCA_ERROR_IO_FAILED - struct `doca_event` { .result.u64 } should contain IPSec CTX specific error status code.

```c
struct doca_job doca_ipsec_sa_create_job::base
```

`doca job object`

```c
struct doca_ipsec_sa_attrs
doca_ipsec_sa_create_job::sa_attrs
```

`ipsec sa attr`

3.48. **doca_ipsec_sa_destroy_job Struct Reference**

DOCA IPSec SA destroy job.
The result of this job as struct `doca_event` { .result.u64 } should contain SA destroy completion status code.

```c
struct doca_job doca_ipsec_sa_destroy_job::base
```

`doca job object`

```c
doca_ipsec_sa *doca_ipsec_sa_destroy_job::sa
```

`ipsec sa object (from create)`

3.49. **doca_job Struct Reference**

Job structure describes request arguments for service provided by context.

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

**DOCA Job layout**

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

```c
doca_ctx *doca_job::ctx
```

Doca CTX targeted by the job.
int doca_job::flags

Job submission flags (see `enum doca_job_flags`).

int doca_job::type

Defines the type of the job.

doca_data doca_job::user_data

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

3.50. doca_log_registrator

Registers log source on program start.

Logging Management `cppClassifierVisibility: visibility=public` `cppClassifierTemplateModel: =` 

Should be used to register the log source. For example:

DOCA_LOG_REGISTER(dpi)

void foo { DOCA_LOG_INFO("Message"); }

Note:
The macro also takes care of the dtor() logic on teardown.

3.51. doca_pci_bdf Struct Reference

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

3.52. doca_regex_job_search Struct Reference

Data required to dispatch a job to a RegEx engine.

uint8_t doca_regex_job_search::allow_batching

Set this to 1 to allow a RegEx device to choose to aggregate jobs into batches. Batching can improve throughput at the cost of latency. Set this to 0 to force this job to begin executing immediately, this will also force any previously enqueued jobs that have been batched and not
yet dispatched to begin processing. Not all devices will support batching. If a device does not have batching support this flag is ignored.

```c
struct doca_job doca_regex_job_search::base
```
Common job data.

```c
const doca_buf *doca_regex_job_search::buffer
```
Data for the job.

```c
doca_regex_search_result *doca_regex_job_search::result
```
Pointer to where the job response is stored. The caller must ensure this pointer is valid when submitting a job and it must remain valid until a response for the job has been retrieved from the RegEx engine. This object will be the returned via the event.result.ptr field.

```c
uint16_t doca_regex_job_search::rule_group_ids
```
IDs which can be used to select which group of rules are used to process this job. Set each value to a non zero value to enable this feature or 0 to ignore it.

### 3.53. doca_regex_match Struct Reference

Description of a RegEx match

```c
uint32_t doca_regex_match::length
```
Length of matched value.

```c
uint32_t doca_regex_match::match_start
```
Index relative to the start of the job / stream where the match begins

```c
doca_regex_match *doca_regex_match::next
```
Allows matches to be linked together for easy management and iteration

```c
uint32_t doca_regex_match::rule_id
```
ID of rule used to generate this match.
3.54. **doca_regex_search_result Struct Reference**

Result of a RegEx search

```c
uint32_t
doca_regex_search_result::detected_matches
```
Total number of detected matches.

```c
doca_regex_match
*doca_regex_search_result::matches
```
Returned matches. Contains num_matches elements as a linked list.

```c
doca_regex_mempool
*doca_regex_search_result::matches_mempool
```
Memory pool owning the matches.

```c
uint32_t doca_regex_search_result::num_matches
```
Total number of returned matches.

```c
uint64_t doca_regex_search_result::status_flags
```
Response flags. A bit masked field for zero or more status flags. See doca_regex_status_flag.

3.55. **doca_rmax_cpu_affinity_mask Struct Reference**

Data structure to describe CPU mask for doca_rmax internal thread.

```c
doca_rmax_cpu_mask_t
doca_rmax_cpu_affinity_mask::cpu_bits
```
CPU is included in affinity mask if the corresponding bit is set
3.56. **doca_rmax_in_stream_completion**

Struct Reference

Completion returned by input stream describing the incoming packets. Input stream starts to receive packets right after start and attaching any flow.

- **uint32_t**
  - doca_rmax_in_stream_completion::elements_count
    - Number of packets received

- **doca_rmax_in_stream_completion::memblk_ptr_arr**
  - Array of pointers to the beginning of the memory block as configured by input stream create step. The offset between packets inside memory block can be queried by
    - [doca_rmax_in_stream_get_memblk_stride_size](#)
  - **uint32_t**
    - doca_rmax_in_stream_completion::memblk_ptr_arr_len
      - Number of memory blocks placed in memblk_ptr_arr. See
        - [doca_rmax_in_stream_get_memblks_count](#)
  - **uint32_t**
    - doca_rmax_in_stream_completion::seqn_first
      - Sequence number of the first packet
  - **uint64_t**
    - doca_rmax_in_stream_completion::ts_first
      - Time of arrival of the first packet
  - **uint64_t**
    - doca_rmax_in_stream_completion::ts_last
      - Time of arrival of the last packet
3.57. **doca_rmax_stream_error Struct Reference**

Detailed completion error information.

```cpp
int doca_rmax_stream_error::code
```

Raw Rivermax error code

```cpp
const char *doca_rmax_stream_error::message
```

Human-readable error

3.58. **doca_sha_job Struct Reference**

DOCA SHA job definition. -- “struct doca_sha_job” is used for one-shot SHA calculation.

- Its typical usage is: -- construct a job: `struct doca_sha_job job = { .base.type = DOCA_SHA_JOB_SHA1, .req_buf = user_req_buf, .resp_buf = user_resp_buf, .flags = DOCA_SHA_JOB_FLAGS_NONE };` -- submit job: `doa_workq_submit(workq, &job.base);` -- retrieve event: `doa_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE);`

- For `doa_workq_submit()` return code: -- DOCA_SUCCESS: -- The job is submitted successfully. It also means: this submitted source data cannot be freely manipulated until its response is received. -- DOCA_ERROR_INVALID_VALUE: -- Some of the job attribute members use illegal value. For example, response buffer length is < 20bytes for SHA1; request buffer length == 0, and the job type attribute is not supported. -- DOCA_ERROR_NO_MEMORY: -- The job resource is exhausted for now, we need to call progress_retrieve() first to receive response and free job resource, then call job_submit() to try again to submit the same job. -- DOCA_ERROR_BAD_STATE: -- sha_ctx is corrupted now, need reset.

- For `doa_workq_progress_retrieve()` return code: -- DOCA_SUCCESS: -- we get a response from SHA engine. user can utilise doca_job’s user_data field to setup special data to correlate the returned event and the corresponding job. -- DOCA_ERROR_AGAIN: -- In order to get a response, we need to call progress_retrieve() again. -- DOCA_ERROR_IO_FAILED: -- abnormal occurs in the SHA engine hardware queue, sha_ctx and workq need to be re-initialized. -- DOCA_ERROR_INVALID_VALUE: -- received invalid input.

```cpp
struct doca_job doca_sha_job::base
```

Opaque structure.
3.59. **doca_sha_partial_job** Struct

Reference

DOCA SHA_PARTIAL job definition. -- "struct doca_sha_partial_job" is used for stateful SHA calculation. -- Its typical usage for a job composed of 3 segments is: -- get a session handle: doca_sha_partial_session *session; doca_sha_partial_session_create(ctx, workq, &session); -- construct the 1st job: struct doca_sha_partial_job job = { .sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_1st_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_NONE, .session = session, }; -- submit 1st segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 1st event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); The purpose of this call is to make sure the 1st_segment processing is finished before we can continue to send the next segment, because it is necessary to sequentially process all segment for generating correct SHA result. And the "user_resp_buf" at this moment contains garbage values. -- after the DOCA_SUCCESS event of the 1st segment is received, we can continue to submit 2nd segment: -- construct the 2nd job: struct doca_sha_partial_job job = { .sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_2nd_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_NONE, .session = session, }; -- submit 2nd segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 2nd event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); The purpose of this call is also to make sure the 2nd_segment processing is
finished. And the “user_resp_buf” at this moment still contains garbage values. -- after the DOCA_SUCCESS event of the 2nd segment is received, we can continue to submit 3rd/final segment: -- construct the 3rd job: struct doca_sha_partial_job job = {.sha_job.base.type = DOCA_SHA_JOB_SHA1_PARTIAL, .sha_job.req_buf = user_req_buf_of_3rd_segment, .sha_job.resp_buf = user_resp_buf, .sha_job.flags = DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL, .session = session, }; -- submit 3rd segment: doca_workq_submit(workq, &job.sha_job.base); -- retrieve 3rd event: doca_workq_progress_retrieve(workq, &event, DOCA_WORKQ_RETRIEVE_FLAGS_NONE); -- After the DOCA_SUCCESS event of the 3rd segment is received, the whole job processing is done. We can get the expected SHA result from “user_resp_buf”, -- release session: doca_sha_partial_session_destroy(ctx, workq, session); -- During the whole process, please make sure to use the same “session” handle. -- And for the last segment, the “DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL” flag must be set.

-- For doca_workq_submit() return code: -- DOCA_SUCCESS: -- The job is submitted successfully. It also means: this submitted source data cannot be freely manipulated until its response is received. -- DOCA_ERROR_INVALID_VALUE: -- Some of the job attribute members use illegal value. for example, response buffer length is < 20bytes for SHA1; request buffer length == 0, and the job type attribute is not supported. -- DOCA_ERROR_NO_MEMORY: -- The job resource is exhausted for now, we need to call progress_retrieve() first to receive response and free job resource, then call job_submit() to try again to submit the same job. -- DOCA_ERROR_BAD_STATE: -- sha_ctx is corrupted now, need reset.

-- For doca_workq_progress_retrieve() return code: -- DOCA_SUCCESS: -- we get a response from SHA engine. user can utilise doca_job's user_data field to setup special data to correlate the returned event and the corresponding job. -- DOCA_ERROR_AGAIN: -- In order to get a response, we need to call progress_retrieve() again. -- DOCA_ERROR_IO_FAILED: -- abnormal occurs in the SHA engine hardware queue, sha_ctx and workq need to be re-initialized. -- DOCA_ERROR_INVALID_VALUE: -- received invalid input.

Note: -- sha_partial_job session requirement: -- make sure the same doca_sha_partial_session used for all segments of a whole job. -- before 1st segment submission, call doca_sha_partial_session_create() to grab a session handle. -- from the 1st to the last segment submission, always reuse the same session handle. -- after the last segment processing, to prevent a session resource leak, the user must explicitly call doca_sha_partial_session_destroy() to release this session handle. -- The doca_sha_partial_session_destroy() is provided to let user to free session handle at his will. -- If a session handle is released before the whole stateful SHA is finished, or if different handles are used for a stateful SHA, the job submission may fail due to job validity check failure; even the job submission successes, and the engine is not stalled, a wrong SHA result is expected. -- The “session” resource is limited, it is user’s responsibility to make sure all allocated “session” handles are released. -- If “DOCA_SHA_JOB_FLAGS_SHA_PARTIAL_FINAL” is not properly set, the engine will not be stalled, but a wrong SHA result is expected.

-- sha_partial_job segment length requirement: -- only the last segment allows seg-byte-count != multiple-of-64 for sha1 and sha256. For example, for the above example code, the 1st
and 2nd segment byte length must be multiple of 64. -- only the last segment allows seg-byte-count != multiple-of-128 for sha512. -- If the above requirement is not met, job_submission will fail.

doca_sha_partial_session
*doca_sha_partial_job::session

An opaque structure for user. Used to maintain state for stateful SHA calculation.

struct doca_sha_job doca_sha_partial_job::sha_job

A basic sha_job.
Chapter 4. Data Fields

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

A

action
doca_dpi_sig_info

action_descs
doca_flow_pipe_cfg

action_idx
doca_flow_actions

action_type
doca_flow_resource_crypto_cfg

actions
doca_flow_pipe_cfg

address
doca_flow_action_field

aging
doca_flow_monitor

aging_res
doca_flow_grpc_response

align
doca_flow_meta

allow_batching
doca_regex_job_search

antireplay
doca_ipsec_sa_attrs

antireplay_enable
doca_ipsec_sa_antireplay

attr
doca_flow_pipe_cfg

audp_hdr
doca_flow_tun
Data Fields

B
base
doca_compress_job
doca_dma_job_memcpy
doca_sha_job
doca_regex_job_search
doca_ipsec_sa_destroy_job
doca_ipsec_sa_create_job
buffer
doca_regex_job_search

C
cb
doca_flow_cfg
cbs
doca_flow_resource_meter_cfg
doca_flow_monitor
cfg
doca_flow_grpc_pipe_cfg
cir
doca_flow_resource_meter_cfg
doca_flow_monitor
code
doca_rmax_stream_error
cpu_bits
doca_rmax_cpu_affinity_mask
crypto_id
doca_flow_actions
ctx
doca_job

D
decap
doca_flow_actions
detected_matches
doca_regex_search_result
devargs
doca_flow_port_cfg
direction
doca_ipsec_sa_attrs
dst_buff
doca_compress_job
Data Fields

**doqa dma job memcp**

**dst_ip**
- doqa dpi parsing info
- doqa flow encap action
- doqa flow action_descs
- doqa dpi parsing info

**dst_mac**
- doqa flow action_descs
- doqa flow encap action

**dst_port**
- doqa flow action_descs

**E**

**elements**
- doqa flow ordered list

**elements_count**
- doqa rmax in stream completion

**encap**
- doqa flow actions

**entry_id**
- doqa flow grpc_response

**entry_status**
- doqa flow grpc_response

**error**
- doqa flow grpc_response

**esn_enabled**
- doqa ipsec sa_attrs

**esn_overlap**
- doqa ipsec sa_attrs

**esn_overlap_event_arm**
- doqa ipsec sa antireplay

**esp_sn**
- doqa flow tun

**esp_spi**
- doqa flow tun

**eth_type**
- doqa flow action_descs

**ethertype**
- doqa dpi parsing info

**F**

**flags**
- doqa ct cfg
Data Fields

**DOCA Libraries API**

<table>
<thead>
<tr>
<th>Data Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>doca_job</td>
</tr>
<tr>
<td>doca_flow_actions</td>
</tr>
<tr>
<td>doca_sha_job</td>
</tr>
<tr>
<td>doca_flow_monitor</td>
</tr>
<tr>
<td>doca_flow_resource_rss_cfg</td>
</tr>
<tr>
<td>doca_flow_match</td>
</tr>
<tr>
<td>fwd</td>
</tr>
<tr>
<td>doca_flow_grpc_fwd</td>
</tr>
<tr>
<td>doca_flow_resource_crypto_cfg</td>
</tr>
<tr>
<td>gre_key</td>
</tr>
<tr>
<td>doca_flow_tun</td>
</tr>
<tr>
<td>gtp_teid</td>
</tr>
<tr>
<td>doca_flow_tun</td>
</tr>
<tr>
<td>hard_lifetime_arm</td>
</tr>
<tr>
<td>doca_ipsec_sa_antireplay</td>
</tr>
<tr>
<td>has_encap</td>
</tr>
<tr>
<td>doca_flow_actions</td>
</tr>
<tr>
<td>header_type</td>
</tr>
<tr>
<td>doca_flow_resource_crypto_cfg</td>
</tr>
<tr>
<td>ib_dev</td>
</tr>
<tr>
<td>doca_ct_cfg</td>
</tr>
<tr>
<td>ib_pd</td>
</tr>
<tr>
<td>doca_ct_cfg</td>
</tr>
<tr>
<td>icv_length</td>
</tr>
<tr>
<td>doca_ipsec_sa_attrs</td>
</tr>
<tr>
<td>idx</td>
</tr>
<tr>
<td>doca_flow_fwd</td>
</tr>
<tr>
<td>doca_flow_ordered_list</td>
</tr>
<tr>
<td>implicit_iv</td>
</tr>
<tr>
<td>doca_encryption_key</td>
</tr>
<tr>
<td>in_dst_ip</td>
</tr>
<tr>
<td>doca_flow_match</td>
</tr>
<tr>
<td>in_dst_mac</td>
</tr>
<tr>
<td>doca_flow_match</td>
</tr>
<tr>
<td>in_dst_port</td>
</tr>
<tr>
<td>doca_flow_match</td>
</tr>
</tbody>
</table>
in_eth_type
   doca_flow_match
in_l4_type
   doca_flow_match
in_src_ip
   doca_flow_match
in_src_mac
   doca_flow_match
in_src_port
   doca_flow_match
in_tcp_flags
   doca_flow_match
in_vlan_tci
   doca_flow_match
info
   doca_dpi_result
   doca_dpi_grpc_result
ipsec_syndrome
   doca_flow_meta
ipv4
   doca_dpi_parsing_info
ipv4_addr
   doca_flow_ip_addr
ipv6
   doca_dpi_parsing_info
ipv6_addr
   doca_flow_ip_addr
is_root
   doca_flow_pipe_attr
K
key
   doca_flow_resource_crypto_cfg
   doca_ipsec_sa_attrs
key_present
   doca_flow_tun
key_sz
   doca_flow_resource_crypto_cfg
L
l4_dport
   doca_dpi_parsing_info
Data Fields

l4_protocol
doca_dpi_parsing_info

l4_sport
doca_dpi_parsing_info

lag_port
doca_flow_meta

length
doca_regex_match

mark
doca_flow_meta

match
doca_flow_pipe_cfg

match_mask
doca_flow_pipe_cfg

match_start
doca_regex_match

matched
doca_dpi_result
doca_dpi_grpc_result

matches
doca_regex_search_result

matches_mempool
doca_regex_search_result

max_packets_per_queue
doca_dpi_config_t

max_sig_match_len
doca_dpi_config_t

memblk_ptr_arr
doca_rmax_in_stream_completion

memblk_ptr_arr_len
doca_rmax_in_stream_completion

message
doca_flow_error
doca_rmax_stream_error

meta
doca_flow_actions
doca_flow_action_descs
doca_flow_match

mod_dst_ip
doca_flow_actions
**mod_dst_mac**
  *doca_flow_actions*

**mod_dst_port**
  *doca_flow_actions*

**mod_src_ip**
  *doca_flow_actions*

**mod_src_mac**
  *doca_flow_actions*

**mod_src_port**
  *doca_flow_actions*

**mod_vlan_id**
  *doca_flow_actions*

**mode**
  *doca_ipsec_sa_attrs*

**mode_args**
  *doca_flow_cfg*

**monitor**
  *doca_flow_pipe_cfg*

**N**

**name**
  *doca_dpi_sig_data*
  *doca_flow_pipe_attr*

**nb_actions**
  *doca_flow_pipe_attr*

**nb_counters**
  *doca_flow_resources*

**nb_entries_processed**
  *doca_flow_grpc_response*

**nb_flows**
  *doca_flow_pipe_attr*

**nb_http_parser_based**
  *doca_dpi_stat_info*

**nb_matches**
  *doca_dpi_stat_info*

**nb_meters**
  *doca_flow_resources*

**nb_ordered_lists**
  *doca_flow_pipe_attr*

**nb_other_l4**
  *doca_dpi_stat_info*

**nb_other_l7**
  *doca_dpi_stat_info*
Data Fields

nb_queues
    doca_ct_cfg
    doca_dpi_config_t

nb_scanned_pkts
    doca_dpi_stat_info

nb_ssl_parser_based
    doca_dpi_stat_info

nb_tcp_based
    doca_dpi_stat_info

nb_udp_based
    doca_dpi_stat_info

net_type
    doca_flow_resource_crypto_cfg

next
    doca_regex_match

next_pipe
    doca_flow_fwd

next_pipe_id
    doca_flow_grpc_fwd

nisp_hdr
    doca_flow_tun

nisp_syndrome
    doca_flow_meta

nr_queues
    doca_flow_resource_rss_cfg

nr_shared_resources
    doca_flow_cfg

num_matches
    doca_regex_search_result

num_of_queues
    doca_flow_fwd

0

offload
    doca_ipsec_sa_attrs

offset
    doca_flow_action_field

ordered_list_pipe
    doca_flow_fwd

ordered_lists
    doca_flow_pipe_cfg

out_dst_ip
    doca_flow_match
out_dst_mac
doca_flow_match

out_dst_port
doca_flow_match

out_eth_type
doca_flow_match

out_l4_type
doca_flow_match

out_src_ip
doca_flow_match

out_src_mac
doca_flow_match

out_src_port
doca_flow_match

out_tcp_flags
doca_flow_match

out_vlan_tci
doca_flow_match

output_chksum
doca_compress_job

P

pipe
doca_flow_fwd

pipe_id
doca_flow_grpc_response
doca_flow_grpcBindableObj

pkt
doca_dpi_result
doca_dpi_grpc_result

pkt_meta
doca_flow_meta
doca_flow_actionDescsMeta

port
doca_flow_pipe_cfg

port_id
doca_flow_grpc_pipe_cfg
doca_flow_port_cfg
doca_flow_fwd
doca_flow_grpcBindableObj

port_meta
doca_flow_meta
priv_data_size
   doca_flow_port_cfg
proto_type
   doca_flow_actions
   doca_flow_resource_crypto_cfg
protocol
   doca_flow_tun
   doca_ipsec_sa_attrs

Q
queue_depth
   doca_flow_cfg
queues
   doca_flow_cfg
queues_array
   doca_flow_resource_rss_cfg

R
raw_key
   doca_encryption_key
reformat_data
   doca_flow_resource_crypto_cfg
reformat_data_sz
   doca_flow_resource_crypto_cfg
reformat_type
   doca_flow_resource_crypto_cfg
remove_flow_enable
   doca_ipsec_sa_antireplay
remove_flow_packet_count
   doca_ipsec_sa_antireplay
remove_flow_soft_lifetime
   doca_ipsec_sa_antireplay
replay_win_state
   doca_ipsec_sa_antireplay
replay_win_sz
   doca_ipsec_sa_antireplay
req_buf
   doca_sha_job
resource
   doca_flow_cfg
resp_buf
   doca_sha_job
result
doca_event
doca_regex_job_search
doca_dma_memcpy_result

rss_flags
doca_flow_fwd

rss_queues
doca_flow_fwd

rule_group_ids
doca_regex_job_search

rule_id
doca_regex_match

S

sa
doca_ipsec_sa_destroy_job

sa_attrs
doca_ipsec_sa_create_job

salt
doca_encryption_key

security
doca_flow_actions

security_ctx
doca_flow_resource_crypto_cfg

seg_len
doca_dpi_grpc_generic_packet

segment
doca_dpi_grpc_generic_packet

seqn_first
doca_rmax_in_stream_completion

server_address
doca_dpi_config_t

session
doca_sha_partial_job

sha_job
doca_sha_partial_job

shared_counter_id
doca_flow_monitor

shared_meter_id
doca_flow_monitor

shared_rss_id
doca_flow_fwd
sig_id
   doca_dpi_sig_data
   doca_dpi_sig_info
size
   doca_flow_ordered_list
soft_lifetime_arm
   doca_ipsec_sa_antireplay
spi
   doca_ipsec_sa_attrs
src_buff
   doca_compress_job
   doca_dma_job_memcpy
src_ip
   doca_dpi_parsing_info
   doca_flow_encap_action
   doca_flow_action_descs
src_mac
   doca_flow_encap_action
   doca_flow_action_descs
src_port
   doca_flow_action_descs
status_flags
   doca_dpi_result
   doca_dpi_grpc_result
   doca_regex_search_result
success
   doca_flow_grpc_response
switch_port_id
   doca_flow_grpc_response

T

total_bytes
   doca_flow_query
total_pkts
   doca_flow_query
ts_first
   doca_rmax_in_stream_completion
ts_last
   doca_rmax_in_stream_completion
ttl
   doca_flow_actions
   doca_flow_action_descs
Data Fields

**tun**
- doca_flow_encap_action
- doca_flow_match

**tunnel**
- doca_flow_action_descs

**type**
- doca_flow_port_cfg
- doca_flow_error
- doca_event
- doca_job
- doca_flow_grpc_bindable_obj
- doca_encryption_key
- doca_flow_pipe_attr
- doca_flow_tun
- doca_flow_ip_addr
- doca_flow_action_desc
- doca_flow_fwd
- doca_flow_meta

**U**
- u32
  - doca_flow_meta
  - doca_flow_action_descs_meta

**unbind_cb**
- doca_flow_cfg

**user_data**
- doca_dpi_grpc_result
- doca_dpi_result
- doca_event
- doca_job
- doca_flow_monitor
- doca_flow_aged_query

**V**
- vlan_id
  - doca_flow_action_descs
- vlan_tci
  - doca_flow_encap_action
- vxlan_tun_id
  - doca_flow_tun
Z
zone
doca_flow_meta
Notice

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

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

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

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

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

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

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

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

Trademarks

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

Copyright

© 2022 NVIDIA Corporation & affiliates. All rights reserved.