



Mellanox Adapters Programmer's Reference Manual (PRM)

Supporting Connect-IB[®], ConnectX[®]-4, ConnectX[®]-4 Lx, ConnectX[®]-5, ConnectX[®]-6, ConnectX[®]-6 Dx, BlueField and BlueField[®]-2

Rev 0.53

NOTE:

THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT □PRODUCT(S)™ AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES AS-IS, WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.



Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

© Copyright 2020. Mellanox Technologies Ltd. All Rights Reserved.

Mellanox®, Mellanox logo, ASAP2 - Accelerated Switch and Packet Processing®, BlueField®, BlueOS®, CloudX logo, Connect - IB®, ConnectX®, CORE-Direct®, GPUDirect®, HPC-X®, LinkX®, Mellanox CloudX®, Mellanox HostDirect®, Mellanox Multi-Host®, Mellanox NEO®, Mellanox NVMeDirect®, Mellanox OpenCloud®, Mellanox OpenHPC®, Mellanox PeerDirect®, Mellanox ScalableHPC®, Mellanox Socket Direct®, PeerDirect ASYNC®, SocketXtreme®, StoreX®, UCX®, UCX Unified Communication X®, UFM®, Unbreakable -Link®, and Virtual Protocol Interconnect® are registered trademarks of Mellanox Technologies, Ltd.

For the complete and most updated list of Mellanox trademarks, visit <http://www.mellanox.com/page/trademarks>.

All other trademarks are property of their respective owners.

Table of Contents

Table of Contents	3
Revision History	41
About this Manual	156
Chapter 1 Introduction	162
1.1 Major Features Summary	162
1.2 Adapter Device Architecture Layers	164
1.2.1 Wire Protocol	166
1.3 Hardware Interfaces	167
1.3.1 Host Interface - PCI Express	168
1.3.1.1 PCI Express	168
1.3.1.2 NC-SI (DMTF-Compliant Management Link)	168
1.3.1.3 I2C-Compatible	168
1.3.2 Network Interface	168
Chapter 2 Operational Overview	170
2.1 Operation Interfaces	170
2.2 Software Interface	171
2.3 HCA Configuration	171
2.4 HCA Operation	171
2.4.1 Send Operation Overview	172
2.4.2 Receive Operation Overview	173
2.4.3 Queue Pairs	174
2.4.3.1 Special QPs	175
2.4.4 Flow Interface Isolation and Protection	176
2.4.4.1 Raw Network Interface – Mellanox Transport	176
2.4.5 Work Queues	176
2.4.5.1 Shared Receive Queue Overview	176
2.4.6 Completion Queues and Completion Events	177
2.4.7 Event Queues	177
2.4.7.1 Interrupts	177
2.4.8 HCA Memory Resources	178
2.5 Virtualization	179
2.6 Quality of Service	179
Chapter 3 Network Ports	180
3.1 Port Model Overview	180
3.2 Physical Port	180
3.2.1 Overview	180

3.2.2	Port Speed Capabilities	180
3.2.3	Port Numbering	182
3.2.4	Congestion Control	182
3.2.4.1	QCN Congestion Control	182
3.2.4.2	RoCEv2 Congestion Control	182
3.2.5	Port States	183
3.2.6	Port Up/Down Flow	183
3.2.7	Flow Control	183
3.2.8	Buffer Management	183
3.2.9	Cable Management	183
3.2.10	Port Control Registers	183
3.2.11	Mellanox Unbreakable Link Technology	183
3.2.12	FEC/RS-FEC	184
3.2.13	LLR/PLR	184
3.2.14	Communication Block	184
3.2.15	Frequency Based DME	184
3.2.16	Width Tuning	184
3.2.17	MLPN	184
3.2.18	Error Generator	184
3.2.19	PRBS Checker/Generator	184

Chapter 4 Networking and Stateless Offloads 185

4.1	Networking Transport Objects	185
4.1.1	Overview	185
4.2	Networking Services Usage	187
4.2.1	Ethernet	187
4.3	Stateless Offloads	188
4.3.1	Checksum Offload	189
4.3.2	Large Send Offload (LSO)	190
4.3.3	Software Parsing - SWP	191
4.3.4	Multi-Packet Send WQE Offload	191
4.3.4.1	Multi-Packet Send WQE Format	191
4.3.4.2	Multi-Packet Send WQE Limitations	192
4.3.5	Enhanced Multi-Packet Send WQE Offload	193
4.3.5.1	Enhanced Multi-Packet Send WQE Format	193
4.3.6	Receive Side Scaling (RSS)	194
4.3.6.1	TIR Spreading Traffic Mechanism	194
4.3.7	Transmit Side Scaling	194
4.3.8	Interrupt Moderation	195
4.3.9	Large Receive Offload (LRO)	195
4.3.9.1	LRO Introduction and Device Capabilities	195
4.3.9.2	LRO Session Creation	195
4.3.9.3	LRO Session Termination	196

4.3.9.4	Merging New Segments Into Existing LRO Session	197
4.3.9.5	LRO Packet Scatter to Memory	198
4.3.9.6	LRO CQE Fields Summary	198
4.3.10	VLAN Insertion/Stripping	201
4.3.11	Header Data Split	201
4.3.12	Hypervisor Offloading for Tunneling.	201
4.3.12.1	Configuration	201
4.3.12.2	Indication in CQE for Tunneling Packet	202
4.3.12.3	RX Hash (RSS) for Tunneling Packets	202
4.3.12.4	Flow Rules for Tunneling Packets	203
4.3.12.5	Checksum	203
4.3.12.6	LSO	204
4.3.12.7	VLAN Stripping/Insertion	204
4.3.12.8	Header Data Split.	205
4.3.12.9	LRO	205
4.3.13	Packet Padding (TX)	205
4.3.14	Start of Packet Padding (RX)	205
4.3.15	End of Packet Padding (RX)	205
4.3.16	Trailer Insertion	206
4.4	Self-Loopback Control Using Transport Domains	206
4.5	Sniffer	207
4.6	ULP Stateless Offloads	209
4.6.1	ULP Stateless Offloads QP Modes	209
4.6.1.1	Basic ULP Offloads	209
4.6.1.2	Enhanced ULP Offloads.	210
4.7	MLNX ETHERTYPE TAGGING	214
4.8	Hairpin	215
4.8.1	Hairpin Creation Flow	215
4.8.2	Destroy Hairpin Flow	216
4.9	TLS Offload	216
4.9.1	Initialization and enable of offload	216
4.9.1.1	Overview	216
4.9.1.2	Detailed Flow	217
4.9.2	TLS Send Flow	218
4.9.2.1	Out of Order - Resync Flow.	218
4.9.3	TLS Receive Flow	219
4.9.3.1	Resync Flow	219
4.9.4	TLS Teardown	220
4.10	NVMEoTCP Offload.	220
4.10.1	Overview	220
4.10.2	CRC offload overview	221
4.10.3	Context Setup Description	221

4.10.3.1 Detailed flow	221
4.10.3.2 Receive operation	222
4.10.3.3 Resync flow	222
4.11 IPSEC Offload	223
4.12 DPP Offload - Direct Packet Placement Offload.	223
4.12.1 Scattering Overview	224
4.12.2 DPP Offload Creation	224
Chapter 5 PCI Interface	226
5.1 PCIe Compliance	226
5.2 Capabilities Reporting	227
5.3 VSC - Vendor Specific Capability	227
5.3.1 Configuration Domains.	230
5.3.1.1 ICMD Space	230
5.3.1.2 CRSPACE Space	230
5.3.1.3 All ICMD Space.	230
5.3.1.4 Expansion ROM	230
5.3.1.5 Nondestructive (ND) CRSPACE Space.	230
5.3.1.6 Scan CRSPACE Space	230
5.3.1.7 NODNIC Initialization Segment.	231
5.3.1.8 Global Semaphores Space.	231
5.3.1.9 MAC Space	231
5.3.2 Gateway Ownership	233
5.3.3 Gateway Read	233
5.3.4 Gateway Write	233
5.4 Initialization Segment.	234
5.5 Data Interface	238
5.5.1 PCIe Attributes	238
Chapter 6 Memory Resources and Utilization.	239
6.1 Interconnect Context Memory	239
6.2 ICM Management in a Virtual Environment.	239
6.3 Memory Resources.	240
6.3.1 Address Translation Tables.	240
6.4 HCA Control Objects (Contexts)	240
6.4.1 Work Queues.	241
6.4.2 Shared Receive Queue Context	242
6.4.3 Completion Queues	242
6.4.4 Event Queues	242
6.5 User Access Region.	242
Chapter 7 Initialization and Teardown	243
7.1 Initialization.	243
7.1.1 Stage 1 - Device Boot From Attached NVMEM.	243

7.1.2	Stage 2 - PCI Device Initialization via Boot Software Enumeration.	243
7.2	HCA Driver Start-up	244
7.3	HCA Driver Teardown and Re-initialization	245
7.3.1	Fast Teardown.	245
7.4	Physical port Initialization and Configuration	246
7.5	Wake On LAN (WOL)	246
7.6	Reset On LAN (ROL)	246
7.7	ICM Management.	246
7.7.1	Trust Model	247
Chapter 8	Software Interface	248
8.1	ISSI - Interface Step Sequence ID	248
8.1.1	ISSI History.	249
8.1.1.1	ISSI = 1.	249
8.2	User Access Region (UAR)	249
8.2.1	UAR Sections	250
8.2.2	UAR Page Format	250
8.2.2.1	Blue Flame	253
8.2.2.2	Sharing UARs	254
8.3	Queue Pairs.	254
8.3.1	Opening a Queue Pair.	254
8.3.2	Modifying the Queue Pair Context	255
8.3.3	Path Migration.	255
8.3.4	Special QPs.	255
8.3.4.1	Configuration and Ports De-multiplexing	255
8.3.4.2	Special QPs - Receive.	255
8.3.4.3	Mellanox Transport	256
8.3.4.4	Raw Ethernet Queue Pair	256
8.3.4.5	Multi User QP.	256
8.3.5	Queue Pair Context Format	258
8.4	RDB Context	278
8.5	DCT - Dynamically Connected Target	279
8.5.1	DCT Context.	279
8.6	Esw_Vport Context - E-switch Virtual Port Context.	284
8.7	NIC_Vport Context - NIC Virtual Port Context	287
8.8	HCA_Vport Context - HCA Virtual Port Context.	295
8.9	Work Queues	298
8.9.1	Work Queues Structure and Access	299
8.9.1.1	Send Queue	300
8.9.1.2	Receive Queue	302
8.9.1.3	Striding Receive WQ	304
8.9.1.4	Shared Receive Queue	306

8.9.1.5 [EMC] Dual Write Queue	307
8.9.2 Doorbell Record	308
8.9.3 WQE Ownership	309
8.9.3.1 Posting a Work Request to Work Queue	310
8.9.3.2 Posting Work Request on Shared Receive Queue	310
8.9.4 Work Request (WQE) Formats	310
8.9.4.1 Send WQE Format	310
8.9.4.2 Send WQE Construction Summary	354
8.9.4.3 Receive WQE Format	360
8.9.4.4 Shared Receive WQE Format	361
8.9.5 UD Address Vector Format	362
8.10 Shared Receive Queue (SRQ)	365
8.10.1 SRQs and QPs	366
8.10.2 SRQ Context	367
8.10.3 Creating and Destroying SRQs	368
8.10.4 Attaching and Detaching QPs to/from SRQs	368
8.10.5 Posting WQEs to SRQ	369
8.10.6 SRQ Completion Semantics	369
8.10.7 SRQ Limit Water Mark Notification	369
8.10.8 Resizing SRQ	369
8.10.9 Contiguous SRQ	369
8.10.10 SRQ Error Semantics	370
8.11 XRC Shared Receive Queues	370
8.11.1 XRC SRQ Context	370
8.12 Transport Interface Receive (TIR)	373
8.13 Transport Interface Send (TIS)	381
8.14 Receive Queue (RQ)	383
8.14.1 RQ States Summary	397
8.14.2 RQ Error Semantics	398
8.14.2.1 Memory RQ with Inline Memory Queue	398
8.14.2.2 RQ Associated with RMP	398
8.14.3 Drop-Less RQ	398
8.15 RQ Table (RQT)	399
8.16 Send Queue (SQ)	402
8.16.1 SQ States Summary	409
8.16.2 SQ Error Semantics	409
8.16.3 Send WQE Inline Header	410
8.16.3.1 Inline Modes	410
8.17 Receive Memory Pool (RMP)	411
8.17.1 RMP States Summary	414
8.18 Flow Table	414
8.18.1 Position in Processing Flow	415

8.18.2 General Structure	415
8.18.2.1 Match Criteria	415
8.18.2.2 Processing Actions	416
8.18.2.3 Flow Table Chaining	417
8.18.2.4 Multi-Processing Paths	417
8.18.2.5 Default Behavior	418
8.18.2.6 Invalid Flows	418
8.18.2.7 Packet Classification Ambiguities	418
8.18.2.8 Flow Statistics	418
8.18.2.9 Flow Tagging	418
8.18.2.10 MetaData Registers	419
8.18.2.11 Special Flow Tables	419
8.18.3 Configuration Interface	420
8.18.3.1 Allocating a Flow Table	420
8.18.3.2 Selecting the Root Flow Table	420
8.18.3.3 Flow Table Level	421
8.18.3.4 Allocating a Match Criteria - Flow Groups	421
8.18.3.5 Allocating a Flow Counter	422
8.18.3.6 Allocating a Packet Reformat Context	422
8.18.3.7 Encapsulation Entropy	423
8.18.3.8 Allocating a Header Modification Context	423
8.18.3.9 Adding a Flow	424
8.18.3.10 Redefining a Flow	426
8.18.3.11 Freeing Resources	427
8.18.3.12 Querying Device Database	427
8.18.3.13 Special Flows Definition	428
8.18.4 Characteristics of Flow Table Types	428
8.18.4.1 NIC Receive	428
8.18.4.2 NIC Receive RDMA	428
8.18.4.3 NIC Receive Sniffer	429
8.18.4.4 NIC Transmit	429
8.18.4.5 NIC Transmit RDMA	429
8.18.4.6 NIC Transmit Sniffer	430
8.18.4.7 E-Switch Forwarding Data Base (FDB)	430
8.18.4.8 E-Switch Ingress Access Control List (ACL)	431
8.18.4.9 E-Switch Egress Access Control List (ACL)	432
8.19 Completion Queues	432
8.19.1 Completion Queue Buffer	433
8.19.1.1 CQE Format	435
8.19.1.2 NVMETCP Extra 64Bytes CQE	447
8.19.1.3 CQE Format For Application Offload	448
8.19.2 CQ DoorBell Record	449
8.19.3 Poll for Completion	450
8.19.3.1 CQE Ownership	450

8.19.3.2 Reporting Completions	451
8.19.4 Request Completion Notification	451
8.19.5 Completion Queue Update Error	453
8.19.6 Resizing a CQ.	453
8.19.7 Completion With Error.	455
8.19.8 SIGERR CQE - Signature Error CQE	457
8.19.9 CQE Compressing	459
8.19.9.1 CQE Compressing.	459
8.19.9.2 CQE Compression Block	460
8.19.9.3 Title	460
8.19.9.4 Mini CQE.	460
8.19.9.5 CQE Compressing Configuration	460
8.19.9.6 CQE Compression Block	460
8.19.9.7 Mini CQE Formats	462
8.19.9.8 Consumer/Producer Counter	464
8.19.9.9 Compressed CQEs Fields Extraction.	464
8.19.9.10 CQE Owner Bit	466
8.19.9.11 CQE Compressing and Interrupts	466
8.19.9.12 Limitations	466
8.19.10 Completion Queue Context (CQC)	466
8.19.11 CQ to EQ Remapping	471
8.19.12 CQE Timestamping	471
8.19.12.1 Conversion to Real Time	471
8.19.12.2 Synchronization with Current Time	471
8.20 Events and Interrupts.	472
8.20.1 Event Queues	472
8.20.2 Event Queue Buffer	472
8.20.2.1 EQE Ownership	473
8.20.2.2 EQE Format	474
8.20.2.3 EQ DoorBell Register	476
8.20.3 Completion Events	476
8.20.4 Polling on EQEs	476
8.20.5 Sharing MSI-X/Interrupt Amongst EQs	476
8.20.6 Event Queue Mapping	476
8.20.7 Completion Events	479
8.20.8 Asynchronous Events and Errors.	480
8.20.8.1 DCT Events	480
8.20.8.2 QP/SQ/RQ/SRQ Events	480
8.20.9 Object Change Event	481
8.20.10 XRQ Error Event	482
8.20.10.1 Page Fault Event.	483
8.20.10.2 Completion Queue Error Event.	487
8.20.11 Debug Events	487
8.20.11.1 DROPPED_PACKET_LOGGED Event	487

8.20.11.2 Driver Tracer Event	488
8.20.11.3 Resource Dump Event	489
8.20.12 Unaffiliated Events and Errors.	490
8.20.12.1 Internal Errors	490
8.20.12.2 Port State Change Event	491
8.20.12.3 GPIO Event	492
8.20.12.4 Port Module Event.	492
8.20.12.5 DCBX Change Event	493
8.20.12.6 PPS Event - Pulse Per Second Event	494
8.20.12.7 Temp Warning Event	495
8.20.12.8 General Notification Event	495
8.20.12.9 Monitor Counter Change Event	499
8.20.13 Performance Related Events	499
8.20.13.1 Doorbell / Blueflame Congestion Event.	499
8.20.13.2 Stall VL Event	500
8.20.14 HCA Interface Events	501
8.20.14.1 Command Interface Completion Event	501
8.20.14.2 Pages Request Event	501
8.20.14.3 NIC Vport Change Event	502
8.20.14.4 ESW_FUNCTIONS_CHANGED Event	503
8.20.15 FPGA Events	504
8.20.15.1 FPGA Error Event	504
8.20.15.2 FPGA QP Error Event	504
8.20.16 Event Queue Context (EQC).	505
8.20.17 Hardware Interrupts.	506
8.20.18 Interrupt Moderation.	507
8.20.19 Completion Event Moderation	508
8.20.20 Interrupt Frequency Moderation	509
8.21 XRQ Context	509
8.21.1 XRQ Context	510
8.21.2 XRQ With Tag Matching Topology	513
8.21.2.1 Matching List	515
8.21.2.2 UWQ - Unexpected WQ	515
8.21.2.3 SW and HW Phase Counters.	515
8.21.2.4 Tag Matching Headers.	516
8.21.3 NVM Express over Fabric (NVMe-oF) Target Offload Context	519
8.22 Transport Static Params Context.	525
8.23 TLS PROGRESS Params Context.	528
8.24 NVMEoTCP Progress Params Context.	529
8.25 FPGA QP	530
8.25.1 FPGA QP States Summary	535
8.26 Transmit Scheduler.	535

8.26.1 Scheduling Arbiter	536
8.26.1.1 Bandwidth Allocation	536
8.26.1.2 Rate Limit	536
8.26.2 Multi-level scheduler	536
8.26.3 Special TSAR Types	537
8.26.3.1 ETS TSAR	537
8.26.3.2 Round Robin TSAR	538
8.26.4 TSAR Configuration.	538
8.26.4.1 TSAR Management	538
8.26.4.2 TSAR Agent Management.	538
8.27 General Object Types	539
8.27.1 UCTX - User Context	539
8.27.2 UMEM - User Memory Context.	540
8.27.3 NVMe Device Emulation Object	541
8.27.4 NVME SQ - NVME Emulation SQ Object	542
8.27.5 NVME_NAMESPACE OBJEC	546
8.27.6 NVME CQ - NVME Emulation CQ Object.	547
8.27.7 NVME_CTRL_COUNTERS Object	549
8.27.8 NVME_NAMESPACE_COUNTERS Object	551
8.27.9 NVME_SQ_COUNTERS Object.	552
8.27.10 SW ICM Block Object	554
8.27.11 Flow Meter Object	556
8.27.12 Flow Sampler Object	560
8.27.13 GENEVE TLV OPTION Object	562
8.27.14 DEK Object	564
8.27.15 IMPORT_KEY Object.	567
8.27.16 CREDENTIAL Object	568
8.27.17 CRYPTO_LOGIN Object.	570
8.27.18 VIRTIO_NET_Q Object	571
8.27.19 VIRTIO NET DEVICE EMULATION Object.	582
8.27.20 VIRTIO BLK DEVICE EMULATION Object	587
8.27.21 VIRTIO_BLK_Q Object	591
8.27.22 HotPlug Device Object	595
8.27.23 NVMF_BACKEND_CONTROLLER Object	600
8.27.24 NVMF_NAMESPACE Object	602
8.27.25 NVMF_NAMESPACE_COUNTER_SET Object	603
8.27.26 VHCA Tunnel Object.	605
8.27.27 IPSEC Offload Object	606
8.27.28 Match Definer Object.	611
8.28 Command Interface	613
8.28.1 HCA Command Queue	613
8.28.1.1 Command Timeout	616
8.28.1.2 Mailbox Format	616

8.28.1.3 Command Data Layout	618
8.29 ICMD Interface	619
8.29.1 ICMD Control	620
8.29.2 Using ICMD Commands	620
8.29.2.1 External Mailbox	621
8.29.3 ICMD Command Formats.	621
8.29.4 ICMD Capabilities	621
Chapter 9 Address Translation and Protection	622
9.1 Virtual to Physical Address Translation and Protection Checks	622
9.1.1 Address Translation Indirection.	624
9.1.2 On-demand Paging Support.	626
9.2 Zero-Based Virtual Address Regions and Windows.	628
9.3 Shared Memory Regions	628
9.4 Response to Memory Access Errors	628
9.5 Reserved LKey.	629
9.6 Address Translation Control Structures	629
9.6.1 MKey Context	629
9.7 Memory Key Configuration (Creation)	634
Chapter 10 Firmware Components Update	635
10.1 Overview	635
10.2 Firmware Components.	635
10.2.1 Component Update State	635
10.2.2 Component Properties.	636
10.3 Update State Machine	636
10.3.1 Update Handle	636
10.3.2 Update Flow	637
10.3.3 Activation Flow	637
10.3.4 Read Flow	638
10.3.5 FSM Monitoring	638
10.3.6 Flow Cancellation	638
10.3.7 Error Handling.	638
10.3.8 FSM States.	639
10.3.8.1 IDLE State	639
10.3.8.2 LOCKED State	640
10.3.8.3 INITIALIZE State	640
10.3.8.4 DOWNLOAD State	640
10.3.8.5 VERIFY State.	640
10.3.8.6 APPLY State	641
10.3.8.7 ACTIVATE State	641
10.3.8.8 UPLOAD/UPLOAD_PENDING State.	641

10.4 Configuration notes	641
Chapter 11 Non-Volatile Configuration (NVConfig)	642
11.1 Configuration Interface	642
11.2 Defaults	643
11.3 NV Configurations Privileges	643
11.4 Lifecycle and Defaults [Unimplemented]	643
11.5 Conventions	644
11.5.1 Classes	644
11.5.2 Naming Guidelines for Mlxconfig	644
11.5.3 Tags, Attributes and Primitives	644
11.6 NVConfig Summary	646
11.6.1 Global Parameters	646
11.6.2 Per Port Parameters	650
11.6.3 Per Host-Function Parameters	652
11.6.4 File Parameters	656
11.6.5 Per Host Parameters	657
11.7 NIC Parameters	657
11.7.1 NV_BASE_MAC_GUID	657
11.7.2 NV_BASE_MAC_GUID_CAP	658
11.7.3 NV_FORBIDDEN_VERSIONS	658
11.7.4 NV_MEMIC_CONF	660
11.7.5 NV_MEMIC_CAP	660
11.7.6 NV_HOST_CHAINING_CONF	661
11.7.7 NV_HOST_CHAINING_CAP	662
11.7.8 NV_FPGA_CONF	663
11.7.9 NV_FPGA_CAP	664
11.7.10 NV_INTERNAL_CPU_CONF	664
11.7.11 NV_INTERNAL_CPU_CAP	665
11.7.12 NV_FLEX_PARSER_CONF	665
11.7.13 NV_FLEX_PARSER_CAP	666
11.7.14 NV_WAKE_ON_LAN_CONF	667
11.7.15 NV_PCI_CONF	668
11.7.16 NV_PCI_CAP	670
11.7.17 SAFE_MODE_CONF	672
11.7.18 NV_HOST_PRIV_CONF	673
11.7.19 NV_HOST_PRIV_CAP	674
11.7.20 NV_WAKE_ON_LAN_CAP	676
11.7.21 NV_VIRT_NET_ADDR	676
11.7.22 NV_VIRT_GUID_ADDR	677
11.7.23 NV_HCA_CONF	678
11.7.24 NV_HCA_CAP	679

11.7.25 NV_ECPF_CONF	679
11.7.26 NV_ECPF_CAP	680
11.7.27 NV_VPI_LINK_TYPE	681
11.7.28 NV_GLOBAL_PCI_CONF	682
11.7.29 NV_GLOBAL_PCI_CAP	686
11.7.30 NV_GLOBAL_PCI_CAP_3	690
11.7.31 NV_GLOBAL_PCI_CONF_2	694
11.7.32 NV_GLOBAL_PCI_CONF_3	695
11.7.33 NV_TPT	701
11.7.34 NV_TPT_CONF	701
11.7.35 NV_SW_OFFLOAD_CONFIG	702
11.7.36 NV_SW_OFFLOAD_CAP	707
11.7.37 NV_PACKET_PACING	710
11.7.38 NV_FILE_ID_MLNX	710
11.7.39 NV_FILE_ID_VENDOR	711
11.7.40 NV_PROG_CC	712
11.7.41 NV_HPCC_INT	713
11.7.42 NV_HPCC_INT_CAP	715
11.7.43 NV_IB_DC_CONF	716
11.7.44 NV_IB_DC_CAP	716
11.7.45 NV_ENCLOSURE_INFO	717
11.7.46 NV_CURRENT_CONFIGURATOR	718
11.7.47 NV_INVALIDATE_ALL_TIMESTAMP	719
11.7.48 NV_DEBUG_MODE	720
11.7.49 NV_LLDP_NB_DCBX	721
11.7.50 NV_LLDP_NB_TX_CHASSIS_ID_TLV	721
11.7.51 NV_LLDP_NB_TX_PORT_ID_TLV	722
11.7.52 NV_LLDP_NB_TX_TTL_TLV	722
11.7.53 NV_LLDP_NB_TX_OPTIONAL_TLV0	723
11.7.54 NV_LLDP_NB_TX_OPTIONAL_TLV1	723
11.7.55 NV_LLDP_NB_TX_OPTIONAL_TLV2	724
11.7.56 NV_LLDP_NB_TX_OPTIONAL_TLV3	724
11.7.57 NV_LLDP_NB_TX_OPTIONAL_TLV4	725
11.7.58 NV_LLDP_NB_TX_OPTIONAL_TLV5	725
11.7.59 NV_LLDP_NB_TX_OPTIONAL_TLV6	726
11.7.60 NV_LLDP_NB_TX_OPTIONAL_TLV7	726
11.7.61 NV_KEEP_LINK_UP	727
11.7.62 NV_QOS_CONF	729
11.7.63 NV_QOS_CAP	730
11.7.64 NV_ROM_IB_DHCP_CONF	731
11.7.65 NV_SECURE_HOST	731

11.7.66 NV_KDNET_DATA	732
11.7.67 NV_PCI_SWITCH_EVB_CONF	733
11.7.68 NV_PCI_SWITCH_EVB_CAP	734
11.7.69 NV_FPGA_DATA	735
11.7.70 NV_EXTERNAL_PORT_CTRL	735
11.7.71 NV_ROCE_1_5_CONF	736
11.7.72 NV_HOST_TO_BMC	737
11.7.73 NV_INTERNAL_RESOURCEHAIRPIN_CONF	737
11.7.74 NV_INTERNAL_HAIRPIN_CAP	738
11.7.75 NV_LLDP_NB_CONF	739
11.7.76 NV_LLDP_NB_CAP	740
11.7.77 NV_ROCE_CC	742
11.7.78 NV_ROCE_CC_ECN	742
11.7.79 NV_ROCE_CC_QCN	746
11.7.80 FILE_APPLICABLE_TO	747
11.7.81 FILE_COMMENT	748
11.7.82 FILE_SIGNATURE	748
11.7.83 FILE_DBG_FW_TOKEN_ID	749
11.7.84 FILE_CS_TOKEN_ID	750
11.7.85 FILE_PUBLIC_KEY	750
11.7.86 FILE_MAC_ADDR_LIST	751
11.7.87 FILE_SIGNATURE_4096_A	752
11.7.88 FILE_SIGNATURE_4096_B	753
11.7.89 FILE_PUBLIC_KEY_4096	753
11.7.90 FILE_PUBLIC_KEY_4096_A	755
11.7.91 FILE_PUBLIC_KEY_4096_B	757
11.7.92 NV_MPFS_CONF	760
11.7.93 NV_MPFS_CAP	762
11.7.94 NV_POWER_CONF	763
11.7.95 NV_POWER_CAP	765
11.7.96 NV_HOST_SYNC_CONF	766
11.7.97 NV_EMULATION_NVME_CONF	767
11.7.98 NV_EMULATION_NVME_CAP	768
11.7.99 NV_EMULATION_PCI_SWITCH_CONF	769
11.7.100 NV_EMULATION_PCI_SWITCH_CAP	770
11.7.101 NV_EMULATION_VIRTIO_NET_CONF	770
11.7.102 NV_EMULATION_VIRTIO_NET_CAP	772
11.7.103 NV_EMULATION_VIRTIO_BLK_CONF	773
11.7.104 NV_EMULATION_VIRTIO_BLK_CAP	775
11.7.105 NV_VTS_BANK_A	776
11.7.106 NV_VTS_BANK_B	776
11.8 Expansion ROM Parameters	778

11.8.1 NV_ROM_INI.....	778
11.8.2 NV_ROM_CAP.....	782
11.8.3 NV_ROM_BOOT_CONF2	784
11.8.4 NV_ROM_UEFI_CONF	784
11.8.5 NV_ROM_BOOT_CONF1	786
11.8.6 NV_ROM_IB_BOOT_CONF.....	789
11.8.7 NV_ROM_ISCSI_GENERAL_CONF	789
11.8.8 NV_ROM_DEBUG_LEVEL.....	791
11.8.9 NV_ROM_UEFI_DEBUG_LEVEL	796
11.8.10 NV_ROM_FLEXBOOT_DEBUG	799
11.8.11 NV_ROM_BANNER_TIMEOUT.....	799
11.8.12 NV_ROM_OCDS_OCBB	800
11.8.13 NV_ROM_ISCSI_DHCP_CONF	800
11.8.14 NV_ROM_ISCSI_DHCP_VENDOR_ID.....	801
11.8.15 NV_ROM_ISCSI_INITIATOR_IP_ADDR.....	801
11.8.16 NV_ROM_ISCSI_INITIATOR_SUBNET_MASK	802
11.8.17 NV_ROM_ISCSI_INITIATOR_GATEWAY	802
11.8.18 NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR.....	803
11.8.19 NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR	803
11.8.20 NV_ROM_ISCSI_INITIATOR_NAME.....	804
11.8.21 NV_ROM_ISCSI_INITIATOR_CHAP_ID.....	804
11.8.22 NV_ROM_ISCSI_INITIATOR_CHAP_PWD	805
11.8.23 NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET.....	805
11.8.24 NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET	806
11.8.25 NV_ROM_ISCSI_FIRST_TARGET_ADDR.....	806
11.8.26 NV_ROM_ISCSI_SECOND_TARGET_ADDR	806
11.8.27 NV_ROM_ISCSI_FIRST_TARGET_PORT	807
11.8.28 NV_ROM_ISCSI_SECOND_TARGET_PORT	807
11.8.29 NV_ROM_ISCSI_FIRST_TARGET_LUN	808
11.8.30 NV_ROM_ISCSI_FIRST_TARGET_LUN	808
11.8.31 NV_ROM_ISCSI_FIRST_TARGET_NAME	809
11.8.32 NV_ROM_ISCSI_SECOND_TARGET_NAME.....	809
11.8.33 NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID	810
11.8.34 NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID	810
11.8.35 NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD	811
11.8.36 NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD	811
11.9 Internal NIC Parameters.....	812
11.9.1 NV_IDX_HM_RESTORE_PARAMS	812
Chapter 12 No-DRAM NIC	813
12.1 Introduction	813
12.2 Limitations.....	813

12.3 Initialization and Teardown	813
12.3.1 Device Initialization	813
12.3.2 Port Initialization	814
12.3.3 Teardown	814
12.3.4 No DRAM NIC Configuration Registers	815
12.3.4.1 General Configurations	815
12.3.4.2 Port Configurations	817
12.3.4.3 Ring Configurations	821
12.4 Setting Receive Filters	822
12.4.1 Promiscuous Mode	822
12.4.2 Promiscuous Multicast Mode	823
12.4.3 Address Filters	823
12.5 Sending Packets	825
12.6 Receiving Packets	826
12.7 Completions and Interrupts	826
12.8 Enabling/Disabling DMA	827
12.9 Event Handling	827
12.10 Working with Other Interfaces	828
12.11 Hardware Formats	828
12.12 Resiliency and Error Recovery	829
12.12.1 Completion Errors	829
12.12.2 Configuration Errors	829
12.12.3 Async Errors	829
Chapter 13 Enhancements Summary	831
13.1 I/O Services' Management and Quality of Service	831
13.1.1 Virtualization Support	831
13.1.2 Send QoS	831
13.1.3 Fast Path for Latency Sensitive Traffic	831
13.2 Network/Transport Enhancements	832
13.2.1 Dynamically Connected Transport Service	832
13.2.2 Enhanced Completion Reporting	832
13.2.2.1 Extended CQE	832
13.2.3 Dual Write	832
13.2.4 Received MADs Processing De-multiplexing	832
13.2.5 Exponential Backoff Timer	832
13.2.5.1 Configuration	833
13.3 Network Application Services	833
13.3.1 Precise Clock Synchronization Over the Network	833
13.3.1.1 Auxiliary Pulse-Per-Second (PPS) Signals	833
13.3.1.2 Congestion Control Improvements	835
13.3.2 Extended Atomic Operations	835

13.3.3 Programmable Offload Engines	835
13.3.4 Programmable Offload Engines	836
13.3.5 Contiguous SRQ	836
13.4 Data Integrity	836
13.4.1 Internal (HW) Data Protection.	836
13.4.2 Control Structures' Integrity Checks	836
13.4.3 Application Level Data Integrity Verification and Signature Handover	836
13.5 Memory Management	836
13.5.1 User Memory Registration.	836
13.5.2 Enforce Ordering on RDMA Read Responses.	837
13.5.3 Enforce Ordering on Gathering Data to be Sent	837
13.5.4 HCA Control Objects' Management	837
Chapter 14 Virtualization.	838
14.1 Ethernet Virtualization	838
14.1.1 Components of Ethernet Virtualization	838
14.1.2 Virtual Ports	839
14.1.2.1 Vport State	840
14.1.2.2 Vport Context.	841
14.1.3 E-switch	841
14.1.3.1 E-switch FDB.	842
14.1.4 UpLink	843
14.1.4.1 ACLs	843
14.1.5 Virtual NIC Start-Up	843
14.1.6 NIC Vport Change Notification	844
14.2 Resource Management for Virtual NICs (vNICs)	844
14.2.1 Trusted virtual NIC	845
14.3 Sub-Function	845
14.4 Multi-port vHCA	845
14.5 Ethernet Virtualization in CPU Devices (BlueField)	846
14.5.1 Model 1 - Separated and Orthogonal Host	847
14.5.2 Model 2 - Embedded CPU (EC)	847
14.5.3 Retrieve Host Information	849
14.6 InfiniBand Network Virtualization.	849
14.6.1 Virtualization Support	850
14.6.1.1 Driver Operation	850
14.6.1.2 SRIOV Controls	850
14.6.2 vHCA Properties and Resources	851
14.6.3 Virtual Ports	851
14.6.3.1 Vport Properties	852
14.6.3.2 Port State	852
14.6.3.3 Identity	853

14.6.3.4 Statistics	854
14.6.4 Special QPs	855
14.6.5 Packet Relay	855
14.6.5.1 Unicast	855
14.6.5.2 Multicast.	855
14.6.6 Asynchronous Events	855
14.6.6.1 Affiliated Events.	855
14.6.6.2 Unaffiliated Events and Errors	855
14.7 vNIC Monitoring	856
14.8 Multi vHCA Mkey	856
14.9 Multi PCI Port	857
14.9.1 Port parent/child vHCA selection	858
14.9.2 Cross vHCA Referencing.	859
14.9.2.1 TIR $\rightarrow$ RQ	859
14.9.2.2 RQT $\rightarrow$ RQ	859
14.9.2.3 MCG $\rightarrow$ QP	860
14.9.3 RX IB Unicast vHCA selection.	860
Chapter 15 LAG - Link Aggregation	861
Chapter 16 Dynamically Connected Transport Service	862
16.1 DC Transport Service Definition	864
16.1.1 DC Packet Format	865
16.1.2 DC Connection and Disconnection Model	865
16.1.2.1 DC Connect and Disconnect	866
16.1.2.2 DC Connection Denial (Connection NAK)	868
16.1.2.3 Lost Disconnect	869
16.1.2.4 Connect Request Reject	870
16.1.2.5 DC Initiator Context State Machine	870
16.1.3 DC Data Packets	872
16.1.4 Data Packets Error Behavior	872
16.2 DC Control Objects	872
16.2.1 DC Target.	872
16.2.2 DC Initiator	874
16.2.3 DC Completion Queue Entries.	874
16.2.3.1 DC Initiator CQE	874
16.2.3.2 DC Target CQE	874
16.2.4 DC CNAK Trace	874
Chapter 17 Data Integrity	877
17.1 Hardware-level Data Integrity	877
17.2 Software-level Data Integrity - Control Objects' Consistency Checks	878
17.2.1 Device Configuration and Control Communication.	878
17.2.2 Memory-resident Control Objects	878
17.2.3 Work Queues Elements Signature	879

17.2.4 Completion and Event Queue Elements (CQEs and EQEs)	879
17.3 Application Data Integrity: Signature Handover Operations	880
17.3.1 Signature Rules and Formats	881
17.3.1.1 T10 DIF Signature Rule	881
17.3.1.2 XOR8 Signature Rule	881
17.3.1.3 XOR32 Signature Rule	881
17.3.1.4 CRC32 Signature Rule	881
17.3.1.5 T10DIF_CS Signature Rule	882
17.3.1.6 CRC32C Signature Rule	882
17.3.1.7 CRC64_XP10 Signature Rule	882
17.3.2 Signature Handover Operation Layer	882
17.3.3 Variable, Repeating, and Transaction Signatures	883
17.3.4 Signature Operation Arguments	883
17.3.5 Signature Handover Operation Steps	884
17.3.6 Signature Check Without Data Transfer and Data Padding	885
17.3.7 Padding	886
17.3.8 Signature Operations' Parameters	886
17.3.8.1 BSF Configuration Notes	889
17.3.8.2 PSV Status Transitions Heuristics	889
17.3.9 Signature Operations Parameter Registration	890
17.3.10 Memory Access - Address and Length Calculation	891
17.3.11 Data and Signature Collocation	893
17.3.12 Data Structure Size and Address Alignment	893
17.3.13 Signatures' Validation and Generation	894
17.3.14 Execution of Signature Handover Operation	894
17.3.14.1 Signature Operation Data Transfer Work Requests	895
17.3.14.2 Signature Handover Completion Report	895
17.3.15 Signature Operation Error Reporting	896
17.3.15.1 Signature Handover Status Synchronization	897
17.3.15.2 Pipelined Status Notifications	898
17.4 Signature Handover Programming	898
17.4.1 Signature Handover Steps	898
17.4.2 Signature Handover Control Structures	899
17.4.2.1 Persistent Signature Value (PSV)	899
17.4.2.2 Byte Stream Format (BSF)	902
17.4.3 Signature Handover Work Requests	913
17.4.3.1 UMR Work Request	913
17.4.3.2 SET_PSV Work Request	913
17.4.3.3 CHECK_PSV Work Request	913
17.4.3.4 GET_PSV Work Request	914
17.4.3.5 RGET_PSV and RCHECK_PSV WQEs	915
17.4.3.6 DUMP Work Request	916

Chapter 18 Address Translation and Protection Enhancements 917

18.1 Byte Stream Format	917
18.2 Dump and Fill Memory Key	917
18.3 Lightweight Memory Registration.	917
18.3.1 User-Mode Memory Registration (UMR)	918
18.3.1.1 Strided Block Format	918
18.3.1.2 UMR Common Programming Model	921
18.3.1.3 UMR Global Pre-Registration Model	921
18.3.1.4 Bind Memory Window Implementation with UMR	922
18.3.1.5 Local Invalidate Implementation with UMR	922
18.3.1.6 UMR Work Request Operation	923
18.4 MEMIC - Memory Mapped to InterConnect	923
18.4.1 MEMIC Allocation	924
18.4.2 MEMIC DE-Allocation	924
18.4.3 Memory Registration	924

Chapter 19 Quality of Service 925

19.1 Transmit Scheduling	925
19.1.1 Embedded Switch Scheduling	925
19.1.1.1 Para-VPort	925
19.1.1.2 Configuration Examples	926
19.1.1.3 Configuration Notes	927
19.2 Packet Pacing	928
19.3 QoS Control Registers	930
19.3.1 QCAP - QoS Capabilities Register	933
19.3.2 QPTS - QoS Priority Trust State Register	934
19.3.3 QDPM - QoS DSCP to Priority Mapping Register	935
19.3.4 QPDPM - QoS Port DSCP to Priority Mapping Register	937
19.3.5 QEPM - QoS EXP to Priority Mapping Register	939
19.3.6 QPCR - QoS Policer Configuration Register	941
19.3.7 QPBR - QoS Policer Binding Register	947
19.3.8 QETCR - QoS ETS TC Configuration Register	949
19.3.9 QEGCS - QoS ETS General Configuration Register	952
19.3.10 QEEC - QoS ETS Element Configuration	954
19.3.11 QPDP - QoS Port Default Priority	959
19.3.12 QPDPC - QoS Port Default PCP	960
19.3.13 QPRT - QoS Priority Regeneration Table	961
19.3.14 QSPTC - QoS Stacking Port TClass Table	962
19.3.15 QTCT - QoS Switch Traffic Class Table	963
19.3.16 QTCTM - QoS Switch Traffic Class Table is MultiCast aware	965
19.3.17 QSTCT - QoS Switch Stacking TClass Table	966
19.3.18 QSPIP - QoS Switch Priority to IEEE Priority Mapping Table	967
19.3.19 QSPCP - QoS Switch Priority to Counting Priority Mapping Table	968
19.3.20 QRWE - QoS ReWrite Enable	969
19.3.21 QPEM - QoS Priority to EXP Mapping Register	969
19.3.22 QPDSM - QoS Priority to DSCP Mapping Register	971

19.3.23 QPPM - QoS Priority to PCP Mapping Register	973
19.3.24 QSL - QoS Switch Lifetime Limit Register	974
19.3.25 QHLL - QoS Head of queue Lifetime Limit Register	976
19.3.26 QCAM - QoS Capabilities Mask Register	977
19.3.27 QPSC - QoS PTP Shaper Configuration Register	978
Chapter 20 Advanced I/O Operation Flows	980
20.1 Cross-channel I/O Operations	980
20.1.1 Using Cross Channel Operations for MPI Collective Offload	984
20.1.2 Cross-channel Operation Work Request Format	984
20.1.2.1 CALC Work Request	984
20.1.2.2 WAIT, SEND_EN and RECEIVE_EN Work Requests	985
20.1.3 Completion and Error Reporting	986
20.2 Vector-Calc Operations	986
20.3 Tag Matching and Rendezvous Offload	986
20.3.1 Overview	986
20.3.2 List Operations	990
20.3.2.1 Append	991
20.3.2.2 Remove	991
20.3.2.3 No operation	992
20.3.3 Tag Matching CQE Generation	992
20.4 Device Emulation	994
20.4.1 Initialization Flow	994
20.4.2 Teardown Flow	995
20.4.3 FLR Flow	995
20.4.4 NVMe Emulation Offload	995
20.4.4.1 Nvme Initialization Flow	996
20.4.4.2 Flow Table configuration	997
20.4.4.3 TearDown Flow	997
20.5 NVM Express over Fabric (NVMe-oF) Target Application Offload	999
20.5.1 NVMe-OF Report Back	1001
20.5.1.1 Increasing NVMF Completion Capsule	1002
20.5.1.2 Report Back Layout	1002
20.6 Disk Encryption	1003
20.6.1 Crypto Concepts	1004
20.6.1.1 Import Methods	1004
20.6.1.2 Import Keys Encrypting Keys (Import_KEK)	1004
20.6.1.3 Data Encryption Keys (DEKs)	1004
20.6.1.4 Credentials	1004
20.6.1.5 Commissioning and Operational Modes	1005
20.6.1.6 MKeys	1005
20.6.1.7 KeyTags	1005
20.6.1.8 Opaque	1006

20.6.2 Crypto Workflow	1006
20.6.2.1 Crypto Capabilities.	1006
20.6.2.2 Switching to Operational Mode	1006
20.6.2.3 Starting a Crypto session.	1007
20.6.2.4 Loading a DEK.	1007
20.6.2.5 Running Data Crypto Operations	1007
20.6.2.6 Updating Import_KEK	1008
20.6.2.7 Updating DEK	1008
20.7 VIRTIO Emulation Offload	1008
20.7.1 Initialization Flow for Offloading VIRTIO_NET Virtqueues	1009
20.7.2 Teardown Flow	1009
20.7.3 Live Migration Flow	1010
20.7.4 Error Flows	1010
Chapter 21 FPGA Device	1011
21.1 Overview	1011
21.2 Initialization and Tear Down Flow.	1011
21.3 FPGA Port	1011
21.4 Software Interface	1012
21.4.1 FPGA QP	1012
21.4.2 FPGA Shell QP Packet Format	1012
21.4.3 FPGA Async Events	1013
21.5 FPGA Sandbox.	1013
21.5.1 Overview	1013
21.5.2 FPGA IPsec Sandbox	1013
21.5.2.1 IPsec Initialization	1013
21.5.2.2 IPsec Teardown	1014
21.5.2.3 IPsec Capabilities	1014
21.5.2.4 IPsec Counters	1016
21.6 MMO - Memory to Memory Offloads.	1018
21.6.1 REGEXP MMO	1019
21.7 IPsec Offload	1022
21.7.1 Overview	1022
21.7.2 Driver Initialization	1022
21.7.3 Key management	1023
21.7.4 Crypto offload for transmitted packets	1023
21.7.4.1 Trailer insertion	1023
21.7.4.2 Checksum Offload	1023
21.7.4.3 LSO Offload.	1024
21.7.4.4 LRO Offload	1024
21.7.4.5 TLS.	1024
21.7.5 Crypto offload for received packets	1024
21.7.6 Full offload for transmitted packets	1024

21.7.6.1 Packet Reformat	1025
21.7.7 Full offload for received packets	1025
21.7.7.1 Packet Reformat	1026
21.7.8 Extended Sequence Number (ESN)	1026
21.7.8.1 ESN update events	1027
21.7.9 IPsec Lifetime	1027
21.7.9.1 Lifetime update Example	1028
21.7.9.2 [Internal] Supporting Lifetime values Larger than 32 bit	1028
21.7.10 Atomic_set(C, real_soft_lifetime)IPsec Syndromes	1029
21.7.11 Limitations	1030
21.7.12 IPsec Decryption for unsupported packets	1030
21.7.12.1 Replay Protection Window High Bound Conditional Update	1031
21.7.12.2 Replay Protection Window conditional update	1033
Chapter 22 CAPI- Coherent Accelerator Programming Interface	1035
22.1 Overview	1035
22.2 CAPI Address Translation	1035
22.3 CAPI Owner	1035
22.4 CAPI Initialization Flow	1035
22.5 CAPI Teardown Flow	1036
Chapter 23 Tunneled Operations	1037
23.1 AS_Notify	1037
23.2 Tunneled atomic:	1038
Chapter 24 Extended Atomic Operations	1039
24.1 Masked Compare and Swap (MskCmpSwap)	1039
24.2 Multifield Fetch and Add (MFetchAdd)	1039
24.3 Packet Format	1040
24.3.1 Short Atomics (1,2 or 4 bytes)	1040
24.4 WQE Format	1041
24.5 RDMA Write Atomicity	1041
Chapter 25 Remote Micro-Applications	1042
Chapter 26 Global Atomicity Guaranteeing (GAG)	1046
Chapter 27 Debug Enhancements	1049
27.1 Dropped Packet Logging	1049
27.2 Health Buffer	1049
27.3 Tracer	1051
27.3.1 Tracer Ownership	1051
27.3.2 Tracing to Memory	1051
27.3.3 Traces Chronology	1052
27.3.3.1 Invalid Timestamp Event Trace	1053

27.3.4 Parsing Traces	1053
27.3.4.1 Timestamp Event Traces	1053
27.3.4.2 String Traces	1055
27.3.4.3 Lost Event Traces	1057
27.3.4.4 Unrecognized Traces	1057
27.3.5 Tracer Initialization	1057
27.3.6 Tracer Teardown	1058
27.4 Diagnostic Counters	1058
27.5 Health Monitoring	1060
27.5.1 RX Path Activity	1060
27.6 Device Health Events	1061
27.7 PCIe Hanged Event	1061
27.8 Resource Dump	1062
27.8.1 Segments	1062
27.8.1.1 Segment Header	1064
27.8.1.2 Segment-Specific Structure	1064
27.8.2 Resource Dump Command Usage	1075
27.8.2.1 Resource Dump Sequence	1076
27.9 Core Dump	1078
27.10 Stress Generation	1078
27.10.1 Stress Types	1079
Chapter 28 Command Registers	1080
28.1 Access Reg Space	1080
28.2 Access Registers	1080
28.2.1 PORT_INFO_RO_FIELDS	1083
28.2.2 NODE_INFO_RO_FIELDS	1083
28.2.3 SET_NODE - Set Node Information	1084
28.3 MISC Registers	1085
28.3.1 POWER_SETTINGS Register	1085
28.3.2 FP_SL_MAP Register	1085
28.3.3 HOST_ENDIANNESS Register	1086
28.3.4 DCBX_PARAM Register	1086
28.3.5 MMHI - Multi-Host Information Register	1089
28.3.6 DCBX_APPLICATION Register	1090
28.3.7 ROCE_ACCL - ROCE Accelerator	1091
28.4 DC Internal Control Register	1093
28.5 Schedule Queue Internal Control Register	1094
28.6 MISC Registers	1094
28.6.1 Loopback Control Register	1094
28.6.2 DB_THRESHOLD Register	1095
28.7 Unit Performance Counters Registers	1096

28.7.1 UNIT_PERF_COUNTER_GLOBAL_CTRL Register	1098
28.7.2 UNIT_PERF_COUNTER_CFG Register	1099
28.7.3 UNIT_PERF_COUNTERS_DATA Register	1101
28.7.4 MISC_COUNTERS Register	1102
28.8 FPGA Registers	1103
28.8.1 FPGA_CAP Register	1103
28.8.2 FPGA_CTRL Register	1107
28.8.3 FPGA_ACCESS_REG_SPACE Register	1111
28.8.4 FPGA_SHELL_COUNTERS Register	1112
28.9 Device Stress Registers	1113
28.9.1 STRS_CAP Register	1113
28.9.2 STRS_STOP_TOGGLE Register	1114
28.9.3 STRS_FAULT_INJECT Register	1122
28.9.4 STRS_MINI_FLOW Register	1124
28.9.5 STRS_RESOURCE Register	1127
28.10 SECURE_HOST	1127
28.11 CORE_DUMP Register	1128
28.12 Resource_DUMP Register	1130
28.13 CAPI Registers	1132
28.13.1 CAPI_CTRL Register	1132
28.14 NCFG - NIC Configuration Register	1134
28.15 CRYPTO_OPERATIONAL Register	1135
28.16 CRYPTO_COMMISSIONING Register	1136
28.17 IMPORT_KEK_HANDLE Register	1136
28.17.1 Usage	1136
28.18 CREDENTIAL_HANDLE Register	1138
28.18.1 Usage	1138
28.19 Network Ports Registers	1140
28.19.1 PCAM - Ports Capabilities Mask Register	1146
28.19.2 PCAP - Ports Capabilities Register	1148
28.19.3 PMLP - Ports Module to Local Port Register	1149
28.19.4 PMTU - Port MTU Register	1151
28.19.5 PTYS - Port Type and Speed Register	1152
28.19.6 PPAD - Port Physical Address Register	1157
28.19.7 PAOS - Ports Administrative and Operational Status Register	1158
28.19.8 PPAOS - Ports Phy Administrative and Operational Status Register . .	1160
28.19.9 PMAOS - Ports Module Administrative and Operational Status Register	1161
28.19.10 [Internal] PMTPS - Port Module Type and Physical Settings Register	1163
28.19.11 PFCC - Ports Flow Control Configuration Register	1165
28.19.12 PPCNT - Ports Performance Counters Register	1171
28.19.12.1 Ethernet IEEE 802.3 Counters	1173
28.19.12.2 Ethernet RFC 2863 Counters	1177
28.19.12.3 Ethernet RFC 2819 Counters	1181

28.19.12.4 Ethernet RFC 3635 Counters	1185
28.19.12.5 Ethernet Extended Counters	1188
28.19.12.6 Ethernet Discard Counters	1191
28.19.12.7 Ethernet Per Priority Counters	1194
28.19.12.8 [Internal] Ethernet Out Of Band Port Counters Data	1198
28.19.12.9 Ethernet Per Traffic Class Counters	1201
28.19.12.10 Ethernet Per Traffic Class Congestion Counters	1202
28.19.12.11 [Internal] Ethernet Per Receive Buffer Counters	1203
28.19.12.12 Link Level Retransmission Counters	1203
28.19.12.13 InfiniBand PortCounters Attribute of the Performance Class Counters	1204
28.19.12.14 InfiniBand Extended PortCounters Attribute of the Performance Class Counters	1206
28.19.12.15 Physical Layer Counters Group	1210
28.19.12.16 USR_XSR Physical Layer Counters Group	1214
28.19.12.17 Physical Layer statistical Counters Group	1217
28.19.12.18 PLR Counters Group	1221
28.19.12.19 [Internal] RS Histogram Counters Group	1223
28.19.12.20 [Internal] ZL FEC Counters Group	1223
28.19.13 PUDE - Port Up/Down Event	1224
28.19.14 PMPE - Port Module Plug/Unplug Event	1226
28.19.15 PMPC - Port Module Present Change	1227
28.19.16 PLIB - Port Local port to InfiniBand and label port mapping Register	1228
28.19.17 PFSC - Port Forward Switching Control	1229
28.19.18 PMMP - Port Module Memory Map Properties Register	1230
28.19.18.1 QSFP Protocol Bytes Override	1232
28.19.18.2 SFP Protocol Bytes Override	1234
28.19.19 PMCR - Port Module Control Register	1237
28.19.20 PGMR - Port And Group Mapping Register	1241
28.19.21 PMDR - Port Mapping Definitions Register	1242
28.19.22 [Internal] PMMIC - Port Module Management Initialization Checker Register	1244
28.19.23 PEPC - Port External Phy Control Register	1246
28.19.24 PPTB - Port Prio To Buffer	1247
28.19.25 PBMC - Port Buffer Management Control Register	1249
28.19.26 PBSR - Port Buffer Status Register	1252
28.19.27 PSPA - Port Switch Partition Allocation	1254
28.19.28 PELC - Port Extended Link Capabilities	1255
28.19.29 PMLPN - Port Mellanox Link Protocol Negotiation	1259
28.19.30 PVLC - Port Virtual Lane Capabilities	1260
28.19.31 PPSC - Port Power Save Control	1262
28.19.32 PLBF - Port Loopback Filter Register	1264
28.19.33 PIFR - Port Isolation Filter Register	1265
28.19.34 PIPG - Port Inter Packet Gap	1266
28.19.35 PPLR - Port Physical Loopback Register	1267
28.19.36 PMPR - Port Module Properties Register	1268
28.19.37 PEIR - Port Error Injection Register	1269
28.19.38 PTER - Port Transmit Errors Register	1270
28.19.39 PLPC - Phy Link Profiles Control	1274
28.19.40 PPLM - Port Phy Link Mode	1277
28.19.41 PCNR- Phy CoNfiguration Register	1285
28.19.42 [Internal] SLRP - Serdes Lane Receive Parameters	1288
28.19.43 [Internal] SLRIP - Serdes Lane Receive Input Buffer Parameters	1298
28.19.44 [Internal] SLRD - Serdes Lane Receive Defect	1300

28.19.45	SLTP - Serdes Lane Transmit Parameters	1302
28.19.46	[Internal] SLUTP - Serdes Lane USR Transmit Parameters	1308
28.19.47	SLRG - Serdes Lane Receive Grade	1310
28.19.48	SLREG - SerDes Lane Receive Eye Grade (Internal)	1315
28.19.49	SLRED - SerDes Lane Receive Eye Diagram	1321
28.19.50	SLCCT - SerDes Lane CTLE Calibration Table Register	1325
28.19.51	[Internal] PTAS - Phy Tuning Algorithm Settings	1327
28.19.52	[Internal] PPLL - Phy PLL Status Register	1330
28.19.53	PHBR - Ports Histogram Binding Register	1334
28.19.53.1	Hist_Parameters for Prio	1336
28.19.54	PHCR - Ports Histogram Configuration Register	1337
28.19.55	PHRR - Ports Histogram Reading Register	1339
28.19.56	PPTT - Port PRBS Transmit Test	1340
28.19.57	PPRT - Port PRBS Receive Test	1346
28.19.58	[Internal] PMPT - Port Module PRBS Test	1352
28.19.59	[Internal] PMPD - Port Module PRBS Diagnostics	1355
28.19.60	PCMR - Ports Check Mask Register	1357
28.19.61	PDDR - Port Diagnostics Database Register	1362
28.19.61.1	phy_manager_link_enabled IB	1367
28.19.61.2	phy_manager_link_enabled Ethernet	1367
28.19.61.3	core_to_phy_link_enabled IB	1368
28.19.61.4	core_to_phy_link_enabled Ethernet	1368
28.19.61.5	cable capability IB	1369
28.19.61.6	cable capability Ethernet	1369
28.19.61.7	Link Active IB	1370
28.19.61.8	Link Active Ethernet	1370
28.19.61.9	Monitor Opcodes	1392
28.19.62	PPBMP - Port Phy BER Monitor Parameters	1404
28.19.62.1	Supported Monitor Parameters per monitor type	1405
28.19.63	PPBMC - Port Phy BER Monitor Control	1406
28.19.64	PPBME - Port Phy BER Monitor Event	1408
28.19.65	[Internal] PHFR - Port Halt Fsm Register	1410
28.19.66	[Internal] PNLB - Ports Negotiation Logger Binding Register	1412
28.19.67	[Internal] PNLR - Ports Negotiation Logger Reading Register	1413
28.19.67.1	AN Logger	1414
28.19.67.2	KR Logger	1415
28.19.68	PGUID- Port GUID Register	1416
28.19.69	PMTM - Port Module Type Mapping Register	1418
28.19.70	[Internal] PLAR - Port Local Allocation Register	1419
28.20	Congestion Control Registers	1421
28.20.1	CWGCR - Congestion WRED ECN General Configuration Register	1423
28.20.2	CWTP - Congestion WRED ECN TClass Profile Register	1425
28.20.3	CWPP - Congestion WRED ECN Pool Profile Register	1427
28.20.4	CWTPM - Congestion WRED ECN TClass and Pool Mapping Register	1428
28.20.5	CPQE - Congestion Phantom Queue ECN Register	1429
28.20.6	CHLTR - Congestion High Latency Table Register	1430
28.20.7	CHLTM - Congestion High Latency Tclass Mapping Register	1431
28.20.8	CHLMM - Congestion High Latency Multicast Mapping Register	1432
28.20.9	CNCT - Congestion Notification Control Register	1433

28.20.10 CPCS - Congestion Point Control and Status Register	1433
28.20.11 CPID - Congestion Point Identifier Register	1435
28.20.12 CNMC - Congestion Notification Message Configuration Register . .	1436
28.20.13 CEGCR - Congestion Elephants Global Configuration Register	1437
28.20.14 CEPC - Congestion Elephants Port Configuration Register	1439
28.20.15 CEDR - Congestion Elephants Detection Register	1440
28.20.16 CEER - Congestion Elephants Entry Register	1441
28.20.17 CWCAM - Congestion WRED Capabilities Mask Register	1447
28.21 Buffer Management Registers	1450
28.21.1 SBGCR - Shared Buffer Global Configuration Register	1452
28.21.2 SBPR - Shared Buffer Pools Register	1453
28.21.3 SBCM - Shared Buffer Class Management Register	1455
28.21.4 SBPM - Shared Buffer Port Management Register	1458
28.21.5 SBMM - Shared Buffer Multicast Management Register	1460
28.21.6 SBME - Shared Buffer Multicast Eligibility Register	1462
28.21.7 SBSR - Shared Buffer Status Register	1463
28.21.8 SBIB - Shared Buffer Internal Buffer Register	1466
28.21.9 SBDCC - Shared Buffer Drop Count CPU	1467
28.21.10 SBDCCM - Shared Buffer Drop Count Multicast	1469
28.21.11 SBDCCR - Shared Buffer Drop Count on Receive	1470
28.21.12 SBHBR - Shared Buffer Histogram Binding Register	1471
28.21.12.1 Hist-Parameters for Port-Tclass	1472
28.21.13 SBHRR - Shared Buffer Histogram Reading Register	1473
28.21.14 SBHBR_V2 - Shared Buffer Histogram Binding Version 2 Register . .	1475
28.21.15 SBHRR_V2 - Shared Buffer Histogram Reading Version 2 Register . .	1478
28.21.16 SBCTC - Shared Buffer Congestion Threshold Configuration Register	1481
28.21.17 SBCTR - Shared Buffer Congestion Threshold Reading Register . . .	1484
28.21.18 SBCTS - Shared Buffer Congestion Threshold Statistics Register . . .	1485
28.21.19 SBHPC - Shared Buffer Headroom Pool Configuration Register . . .	1486
28.21.20 SBSNT - Shared Buffer SNApshot Take Register	1487
28.21.21 SBSNS - Shared Buffer SNApshot Status Register	1488
28.21.22 SBSNTE - Shared Buffer SNApshot Trigger Enable	1489
28.21.23 SBCAM - Shared Buffer Capabilities Mask Register	1490
28.22 Management and Peripherals Control Registers	1492
28.23 Fan Registers	1499
28.23.1 MFCR - Management Fan Control Register	1499
28.23.2 MFSC - Management Fan Speed Control Register	1500
28.23.3 MFSM - Management Fan Speed Measurement	1501
28.23.4 MFSL - Management Fan Speed Limit	1502
28.23.5 FORE - Fan Out of Range Event	1503
28.23.6 MFNR - Management Fan Part and Serial Number Register	1504
28.24 Temperature, Voltage and Power Registers	1505
28.24.1 MTCAP - Management Temperature Capabilities	1505
28.24.2 MTECR - Management Temperature Extended Capabilities Register .	1506
28.24.3 MTMP - Management Temperature	1507
28.24.4 MTWE - Management Temperature Warning Event	1510
28.24.5 MTEWE - Management Temperature Extended Warning Event	1511
28.24.6 MTBR - Management Temperature Bulk Register	1512

28.24.6.1 MTBR Temperature Record	1513
28.24.7 MVCAP - Management Voltage and Current Capabilities.	1514
28.24.8 MVCR - Management Voltage and Current Register	1515
28.24.9 MSPS - Management System Power Supply Register	1516
28.24.10 MCPP - Management Control of Power Policer	1518
28.25 Flash Registers	1520
28.25.1 MFPA - Management Flash Parameters Access	1520
28.25.2 MFBA - Management Flash Block Access.	1522
28.25.3 MFBE - Management Flash Block Erase.	1523
28.25.4 MFMC - Management Flash Miscellaneous Configuration Register . .	1524
28.26 Peripheral Registers.	1526
28.26.1 MCIA - Management Cable Info Access	1526
28.26.2 MCAS - Management Cable Stamping	1530
28.26.3 MCION - Management Cable IO and Notifications Register	1533
28.26.4 [Internal] MQSP - Module QSa Properties Register	1534
28.26.5 MMIA - Management Manufacturing Information Access	1535
28.26.6 MMDIO - Management MDIO Interface	1536
28.26.7 MPGR - Management Power Governor Register	1537
28.26.8 MFM - Management Fabric Memory Register	1538
28.26.9 MHSR - Management Health State Register	1539
28.26.10 MJTAG - Management JTAG Register	1540
28.26.11 MSGI - Management System General Information Register	1542
28.26.12 MSCI - Management System CPLD Information Register.	1543
28.26.13 MPCIR - Management and PHY Control for FW ISSU Flow Register	1544
28.26.14 MGIR - Management General Information Register.	1545
28.26.15 MGPIR - Management General Peripheral Information Register . . .	1555
28.26.16 MDIR - Management Device ID Register	1557
28.26.17 MSSIR - Management System Status Information Register.	1558
28.26.18 MHMPR - Management Host Management of Ports Register.	1560
28.26.19 MRTC - Management Real Time Clock	1561
28.26.20 MCWD – Management CPU Watchdog Register	1562
28.26.21 MPFM – Management PCIe Failure Mode Register	1563
28.26.22 MRSR – Management Reset and Shutdown Register.	1564
28.26.23 MPGO - Management Pulse Generator Output	1565
28.26.24 MLCR – Management LED Control Register	1566
28.26.25 MGCR - Management GPIO Capabilities Register	1567
28.26.26 MERR - Management Entities Remapping Register	1569
28.26.27 MRMR – Management Resource Monitoring Register.	1570
28.27 Non-Volatile Registers.	1571
28.27.1 MNVDA - Non-Volatile Data Access	1571
28.27.2 MNVDI - Non-Volatile Data Invalidate	1578
28.27.3 MNVIA - Non-volatile data Invalidate All.	1579
28.27.4 MNVGC - Non-volatile Global Capabilities.	1580
28.27.5 MNVGN - Non-Volatile Get Next	1581
28.27.6 MGNLE - Management Get Next Log Entry	1583
28.27.7 MNVCT - Non-volatile commit	1585

28.27.8 MNVQC - Non-volatile query capabilities	1586
28.28 Tracer Registers	1587
28.28.1 MTRC_CAP -Tracer Capability Register	1587
28.28.2 MTRC_CONF - Tracer Configuration Register	1589
28.28.3 MTRC_STDB - Tracer String DB Register.....	1590
28.28.4 MTRC_CTRL - Tracer Control Register	1591
28.29 PCIe Registers.....	1593
28.29.1 MPEIN - Management PCIE INfo Register	1593
28.29.2 MPCNT - Management PCIe Counters Register	1597
28.29.2.1 PCIe Performance Counters	1598
28.29.2.2 [Internal] PCIe timers and states	1600
28.29.2.3 [Internal] PCIe lanes counters	1602
28.29.3 MPEGC - Management PCIe General Configuration Register	1604
28.29.4 MPIR - Management PCIe Index Register.....	1606
28.29.5 MPECS - Management PCIe Configuration Space Register	1607
28.29.6 [Internal] MISU – Management Internal Stack Utilization Register ..	1609
28.29.7 MPPF - Management PCIe Physical Functions Register	1610
28.30 Pulse Per Second Registers	1611
28.30.1 MTPPS – Management Pulse Per Second Register	1611
28.30.2 MTPPSE – Management Pulse Per Second Event Register.....	1616
28.30.3 MTUTC – Management UTC Register.....	1617
28.31 Management Component.....	1619
28.31.1 MCQS - Management Component Query Status.....	1619
28.31.2 MCQI - Management Component Query Information	1622
28.31.3 MCC - Management Component Control.....	1629
28.31.4 MCDA - Management Component Data Access	1633
28.31.5 MQIS - Management Query Information Strings.....	1634
28.31.6 MIRC - Management Image Re-activation Control Register.....	1635
28.31.7 MCDD - Management Component DMA Based Data Access	1636
28.31.8 MDFCR - Management Peripheral Devices Firmware Control Register	1637
28.31.9 MDCR - Management Peripheral Devices Control Register.....	1639
28.31.10 MDRCR - Management Peripheral Devices Reset Control Register .	1641
28.31.11 MDSR - Management Debug Session Register	1642
28.32 General Management	1643
28.32.1 MCAM - Management Capabilities Mask Register	1643
28.32.2 MFRL - Management Firmware Reset Level	1645
Chapter 29 Command Reference	1647
29.1 Introduction	1647
29.2 Return Status Summary	1658
29.3 Initialization and General Commands.....	1659
29.3.1 QUERY_PAGES - Query Device Free Memory Pool Status	1660
29.3.2 MANAGE_PAGES - Driver Delivers Memory Pages for the Device Usage or Returns Pages	1662
29.3.3 SET_TUNNELED_OPERATION_PARAMS - Set Tunneled Operation Parameters	1665
29.3.4 QUERY_HCA_CAP – Query Device Capabilities	1667

29.3.4.1 HCA Device Capabilities	1671
29.3.4.2 ODP Capabilities.	1725
29.3.4.3 Atomic Capabilities	1728
29.3.4.4 RoCE Capabilities	1732
29.3.4.5 Networking Offload Capabilities.	1734
29.3.4.6 Flow Table Capabilities	1741
29.3.4.7 E-switch Capabilities	1768
29.3.4.8 Vector Calc Capabilities.	1772
29.3.4.9 Quality of Service Capabilities	1773
29.3.4.10 Debug Capabilities	1777
29.3.4.11 NVMe-oF Capabilities	1779
29.3.4.12 Device Memory Capabilities	1784
29.3.4.13 NVME Device Emulation Capabilities.	1787
29.3.4.14 TLS Capabilities.	1788
29.3.4.15 IPSEC Capabilities.	1789
29.3.4.16 DPP Capabilities	1791
29.3.4.17 VIRTIO Emulation Capabilities	1792
29.3.4.18 Event Capabilities.	1795
29.3.4.19 HotPlug Capabilities.	1798
29.3.4.20 Wqe Based Flow Table Capabilities	1799
29.3.4.21 NVMEoTCP Capabilities.	1800
29.3.4.22 Crypto Capabilities.	1801
29.3.5 SET_HCA_CAP – Set Device Capabilities	1803
29.3.6 QUERY_OTHER_HCA_CAP - Query Other HCA Capabilities	1807
29.3.7 MODIFY_OTHER_HCA_CAP - Modify Other HCA Capabilities.	1809
29.3.8 QUERY_SF_PARTITIONS - Query Sub-Function Partition Configuration	1810
29.3.9 SET_SF_PARTITIONS - Set Sub-Function Partition Configuration	1812
29.3.10 ALLOC_SF - Allocate Sub-Function	1814
29.3.11 DEALLOC_SF - De-Allocate Sub-Function	1815
29.3.12 QUERY_ADAPTER – Query Adapter	1816
29.3.13 INIT_HCA – INIT HCA	1820
29.3.14 TEARDOWN_HCA – Tear-down HCA.	1821
29.3.15 ENABLE_HCA.	1823
29.3.16 DISABLE_HCA	1825
29.3.17 QUERY_ISSI	1826
29.3.18 SET_ISSI	1828
29.3.19 SET_DRIVER_VERSION	1829
29.3.20 QUERY_DRIVER_VERSION	1830
29.3.21 ALLOC_PD - Allocate Protection Domain	1832
29.3.22 DEALLOC_PD - De-Allocate Protection Domain	1833
29.3.23 ALLOC_UAR - Allocate UAR.	1834
29.3.24 DEALLOC_UAR - De-Allocate UAR	1835
29.3.25 ALLOC_XRCD - Allocate XRC Domain	1836

29.3.26	DEALLOC_XRCD - De-allocate XRC Domain	1837
29.3.27	CONFIG_INT_MODERATION - Configure Interrupt Moderation	1838
29.3.28	SET_BURST_SIZE	1840
29.3.29	QUERY_BURST_SIZE	1841
29.3.30	ACTIVATE_TRACER – Activate the tracer	1842
29.3.31	DEACTIVATE_TRACER – Deactivate the tracer	1843
29.3.32	ALLOC_TRANSPORT_DOMAIN - Allocate Transport Domain	1844
29.3.33	DEALLOC_TRANSPORT_DOMAIN - De-Allocate Transport Domain .	1845
29.3.34	SET_WOL_ROL – Set WOL and ROL Mode	1846
29.3.35	QUERY_WOL_ROL – Query WOL and ROL Mode.	1848
29.4	LAG Commands.	1849
29.4.1	CREATE_LAG	1849
29.4.2	MODIFY_LAG.	1851
29.4.3	QUERY_LAG.	1853
29.4.4	DESTROY_LAG	1854
29.4.5	CREATE_VPORT_LAG	1854
29.4.6	DESTROY_VPORT_LAG	1856
29.5	Registers Access Commands	1857
29.5.1	ACCESS_REGISTER	1857
29.5.2	ACCESS_REG_SPACE.	1858
29.5.3	ACCESS_REG_SPACE_DWORD.	1859
29.6	TPT Commands	1861
29.6.1	CREATE_MKEY – Create MKey Entry.	1861
29.6.2	QUERY_MKEY – Query MKey Entry.	1865
29.6.3	DESTROY_MKEY – Destroy MKey Entry	1868
29.6.4	PAGE_FAULT_RESUME.	1869
29.6.5	QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers	1871
29.6.6	ALLOC_MEMIC – Allocate MEMIC.	1873
29.6.7	DEALLOC_MEMIC – De-Allocate MEMIC	1874
29.7	Signature Handover Commands	1876
29.7.1	CREATE_PSV	1876
29.7.2	DESTROY_PSV	1878
29.7.3	QUERY_PSV – Query PSV	1879
29.8	EQ Commands.	1879
29.8.1	CREATE_EQ – Create EQ	1880
29.8.2	DESTROY_EQ – Destroy EQ	1883
29.8.3	QUERY_EQ – Query EQ	1884
29.8.4	GEN_EQE – Generate Event Queue Entry	1886
29.9	CQ Commands	1887
29.9.1	CREATE_CQ – Create Completion Queue	1888
29.9.2	DESTROY_CQ – Destroy CQ	1890
29.9.3	QUERY_CQ – Query CQ	1891

29.9.4 MODIFY_CQ – Modify CQ Parameters	1894
29.10 QP Commands.	1897
29.10.1 CREATE_QP - Create new QP.	1899
29.10.2 DESTROY_QP – Destroying a QP	1904
29.10.3 RST2INIT_QP - Reset to Init QP	1905
29.10.4 INIT2INIT_QP – Init to Init QP	1907
29.10.5 INIT2RTR_QP – Init to RTR QP	1909
29.10.6 RTR2RTS_QP –RTR to RTS QP	1914
29.10.7 RTS2RTS_QP – RTS to RTS QP	1917
29.10.8 SQERR2RTS_QP – SQErr to RTS QP	1922
29.10.9 2ERR_QP – To Error QP	1924
29.10.10 2RST_QP – To Reset QP	1925
29.10.11 QUERY_QP – Query QP	1927
29.10.12 SQD2RTS_QP – SQD to RTS QP	1929
29.10.13 SUSPEND_QP - Suspend a QP	1931
29.10.14 RESUME_QP - Resume a QP	1932
29.10.15 SET_RDB - SET RDB context.	1933
29.10.16 QUERY_RDB - QUERY RDB context	1935
29.10.17 SEND_NVMMF_CC - Send a NVMMF Command Capsule via NVME SQ.	1936
29.10.18 QUERY_NVMMF_CC_RESPONSE - Query NVMMF Command Capsule Response	1938
29.10.19 QUERY_EMULATED_FUNCTIONS_INFO - Query Emulated Functions Information	1940
29.11 REGEXP Commands	1942
29.11.1 SET_REGEXP_PARAMS – Set REGEXP PARAMS	1942
29.11.2 QUERY_REGEXP_PARAMS – QUERY REGEXP PARAMS	1944
29.11.3 SET_REGEXP_REGISTER – Set REGEXP Register	1946
29.11.4 QUERY_REGEXP_REGISTER – QUERY REGEXP Register.	1947
29.12 TIR Commands	1948
29.12.1 CREATE_TIR – Create TIR	1949
29.12.2 MODIFY_TIR – Modify TIR	1950
29.12.3 DESTROY_TIR – Destroy TIR.	1952
29.12.4 QUERY_TIR – Query TIR	1953
29.13 TIS Commands.	1954
29.13.1 CREATE_TIS – Create TIS	1955
29.13.2 MODIFY_TIS – Modify TIS	1956
29.13.3 DESTROY_TIS – Destroy TIS	1958
29.13.4 QUERY_TIS – Query TIS	1959
29.14 Send Queue (SQ) Commands.	1960
29.14.1 CREATE_SQ – Create Send Queue.	1960
29.14.2 MODIFY_SQ – Modify Send Queue.	1962
29.14.3 DESTROY_SQ – Destroy a Send Queue Context.	1963

29.14.4 QUERY_SQ – Query Send Queue	1964
29.15 Receive Queue (RQ) Commands	1966
29.15.1 CREATE_RQ – Create Receive Queue	1966
29.15.2 MODIFY_RQ – Modify Receive Queue	1967
29.15.3 DESTROY_RQ – Destroy a Receive Queue Context	1969
29.15.4 QUERY_RQ – Query Receive Queue	1970
29.15.5 SET_DELAY_DROP_PARAMS – Set Delay Drop Parameters	1971
29.15.6 QUERY_DELAY_DROP_PARAMS – Query Delay Drop Parameters	1972
29.16 RQT Commands	1974
29.16.1 CREATE_RQT – Create RQT	1974
29.16.2 MODIFY_RQT – Modify RQ Table	1975
29.16.3 DESTROY_RQT – Destroy RQT	1977
29.16.4 QUERY_RQT – Query RQT	1978
29.17 Receive Memory Pool (RMP) Commands	1979
29.17.1 CREATE_RMP – Create Receive Memory Pool	1980
29.17.2 MODIFY_RMP – Modify Receive Memory Pool	1981
29.17.3 DESTROY_RMP – Destroy a Receive Memory Pool Context	1983
29.17.4 QUERY_RMP – Query Receive Memory Pool	1984
29.18 Flow Table Commands	1985
29.18.1 CREATE_FLOW_TABLE - Allocate a New Flow Table	1986
29.18.2 MODIFY_FLOW_TABLE - Modify a Flow Table	1991
29.18.3 DESTROY_FLOW_TABLE - De-allocate a Flow Table	1993
29.18.4 SET_FLOW_TABLE_ROOT - Set Flow Table Root	1995
29.18.5 QUERY_FLOW_TABLE - Query Flow Table	1998
29.18.6 CREATE_FLOW_GROUP - Define a New Flow Group	2000
29.18.7 DESTROY_FLOW_GROUP - De-allocate a Flow Group	2018
29.18.8 QUERY_FLOW_GROUP - Query Flow Group	2020
29.18.9 SET_FLOW_TABLE_ENTRY - Set Flow Table Entry	2023
29.18.10 QUERY_FLOW_TABLE_ENTRY - Query Flow Table Entry	2034
29.18.11 DELETE_FLOW_TABLE_ENTRY - Invalidate Flow Table Entry	2036
29.18.12 ALLOC_FLOW_COUNTER - Allocate a Flow Counter	2038
29.18.13 DEALLOC_FLOW_COUNTER - De-Allocate Flow Counter	2040
29.18.14 QUERY_FLOW_COUNTER – Query Flow Counter	2041
29.18.15 ALLOC_PACKET_REFORMAT_CONTEXT - Allocate Packet Reformat Context	2043
29.18.16 DEALLOC_PACKET_REFORMAT_CONTEXT - De-Allocate Packet Reformat Context	2046
29.18.17 QUERY_PACKET_REFORMAT_CONTEXT – Query Packet Reformat Context	2047
29.18.18 ALLOC_MODIFY_HEADER_CONTEXT - Allocate Modify Header Context	2048
29.18.19 DEALLOC_MODIFY_HEADER_CONTEXT - De-Allocate Modify Header Context	2054
29.18.20 QUERY_MODIFY_HEADER_CONTEXT – Query Modify Header Context	2055
29.18.21 SYNC_STEERING – Synchronize Steering Tables	2057
29.19 L2 Table Commands	2058

29.19.1 SET_L2_TABLE_ENTRY - Set L2 Table Entry	2059
29.19.2 QUERY_L2_TABLE_ENTRY - Query L2 Table Entry	2060
29.19.3 DELETE_L2_TABLE_ENTRY - Invalidate Flow Table Entry	2062
29.20 FPGA QP Commands.	2063
29.20.1 FPGA_CREATE_QP - Create FPGA QP	2063
29.20.2 FPGA_MODIFY_QP - Modify FPGA QP	2065
29.20.3 FPGA_QUERY_QP - Query FPGA QP	2066
29.20.4 FPGA_DESTROY_QP - DESTROY FPGA QP.	2067
29.20.5 FPGA_QUERY_QP_COUNTERS - Query FPGA QP Counters	2068
29.21 AFU Command - Acceleration Functions Units commands.	2070
29.21.1 CREATE_AFU Command.	2070
29.21.2 QUERY_AFU Command	2073
29.21.3 DESTROY_AFU Command	2074
29.22 CAPI Commands	2075
29.22.1 CREATE_CAPI_PEC	2075
29.22.2 QUERY_CAPI_PEC	2077
29.22.3 DESTROY_CAPI_PEC	2078
29.23 General Object Commands	2079
29.23.1 CREATE_GENERAL_OBJ	2081
29.23.2 MODIFY_GENERAL_OBJ – Modify General Object	2084
29.23.3 QUERY_GENERAL_OBJ	2085
29.23.4 DESTROY_GENERAL_OBJ	2086
29.23.5 CREATE_UCTX	2087
29.23.6 QUERY_UCTX	2088
29.23.7 DESTROY_UCTX.	2090
29.23.8 CREATE_UMEM	2090
29.23.9 QUERY_UMEM	2093
29.23.10 DESTROY_UMEM	2094
29.24 Special QPs and Management Commands	2095
29.24.1 QUERY_MAD_DEMUX.	2096
29.24.2 SET_MAD_DEMUX	2100
29.24.3 MAD_IFC - Management Datagram Interface	2101
29.25 SRQ Commands.	2103
29.25.1 CREATE_SRQ – Create SRQ	2103
29.25.2 DESTROY_SRQ – Destroy an SRQ	2105
29.25.3 QUERY_SRQ – Query SRQ	2106
29.25.4 ARM_RQ	2108
29.26 XRC_SRQ Commands	2109
29.26.1 CREATE_XRC_SRQ – Create XRC_SRQ.	2110
29.26.2 DESTROY_XRC_SRQ – Destroy an XRC_SRQ.	2112
29.26.3 QUERY_XRC_SRQ – Query XRC_SRQ.	2113

29.26.4 ARM_XRC_SRQ – Arm XRC SRQ	2115
29.27 XRQ Commands	2116
29.27.1 CREATE_XRQ – Create XRQ	2117
29.27.2 DESTROY_XRQ – Destroy an XRQ	2118
29.27.3 QUERY_XRQ – Query XRQ	2119
29.27.4 MODIFY_XRQ – Modify XRQ	2120
29.27.5 QUERY_XRQ_DC_PARAMS_ENTRY – Query XRQ DC PARAMS Entry .	2122
29.27.6 SET_XRQ_DC_PARAMS_ENTRY – SET XRQ DC PARAMS Entry	2124
29.27.7 QUERY_XRQ_ERROR_PARAMS – Query XRQ Error Parameters.....	2125
29.27.8 RELEASE_XRQ_ERROR – Release XRQ Error.....	2128
29.28 NVMe-oF Commands	2129
29.28.1 CREATE_NVMF_BACKEND_CONTROLLER – Create NVMe-oF Backend Controller	2129
29.28.2 DESTROY_NVMF_BACKEND_CONTROLLER – Destroy NVMe-oF Backend Controller	2132
29.28.3 QUERY_NVMF_BACKEND_CONTROLLER – Query NVMe-oF Backend Controller	2133
29.28.4 ATTACH_NVMF_NAMESPACE – Attach NVMe-oF Namespace	2134
29.28.5 DETACH_NVMF_NAMESPACE – Detach NVMe-oF Namespace	2136
29.28.6 QUERY_NVMF_NAMESPACE_CONTEXT – Query NVMF NameSpace Context	2137
29.29 QUERY_ESW_FUNCTIONS - Query e-Switch Functions	2140
29.30 DCT Commands.....	2143
29.30.1 CREATE_DCT – Create DCT	2144
29.30.2 DESTROY_DCT – Destroying a DCT	2145
29.30.3 QUERY_DCT – Query DCT	2146
29.30.4 DRAIN_DCT – Draining a DCT	2147
29.30.5 ARM_DCT_FOR_KEY_VIOLATION – Arms DCT for Key Violation	2148
29.30.6 CREATE_DCT_LAG – Create DCT LAG	2150
29.30.7 SET_DC_CNAK_TRACE – Set DC CNAK Trace	2151
29.30.8 QUERY_DC_CNAK_TRACE – Query DC CNAK Trace.....	2153
29.31 Vport Commands	2154
29.31.1 QUERY_VPORT_STATE – Query Vport State.....	2155
29.31.2 MODIFY_VPORT_STATE – Modify Vport State.....	2157
29.31.3 QUERY_ESW_VPORT_CONTEXT – Query E-switch Vport Context ...	2158
29.31.4 MODIFY_ESW_VPORT_CONTEXT – Modify E-switch Vport Context .	2160
29.31.5 QUERY_NIC_VPORT_CONTEXT – Query NIC Vport Context	2162
29.31.6 MODIFY_NIC_VPORT_CONTEXT – Modify NIC Vport Context	2164
29.31.7 QUERY_HCA_VPORT_CONTEXT – Query HCA Vport Context.....	2166
29.31.8 MODIFY_HCA_VPORT_CONTEXT – Modify HCA Vport Context.....	2167
29.31.9 QUERY_HCA_VPORT_GID – Query HCA VPORT GID Table	2169
29.31.10 QUERY_HCA_VPORT_PKEY – Query HCA Vport Pkey Table	2170
29.32 QUERY_ROCE_ADDRESS – Query RoCE Address Entry	2172
29.33 SET_ROCE_ADDRESS – SET RoCE Address Entry	2175
29.34 Vport CountersMonitoring Commands.....	2176

29.34.1 QUERY_VPORT_COUNTER – Query Vport Counter	2176
29.34.2 QUERY_VPORT_COUNTER – Query Vport Counter	2177
29.34.3 QUERY_VNIC_ENV – Query vNIC Environment	2181
29.35 Q Counters Commands.	2185
29.35.1 ALLOC_Q_COUNTER - Allocate a Q-Counter	2185
29.35.2 DEALLOC_Q_COUNTER - De-Allocate Q-Counter	2187
29.35.3 QUERY_Q_COUNTER – Query Q Counter.	2188
29.35.4 SET_MONITOR_COUNTER - Set Monitor Counter.	2195
29.35.5 ARM_MONITOR_COUNTER - ARM Monitor Counter	2198
29.36 QoS Commands - Quality of Service Commands	2199
29.36.1 SET_PP_RATE_LIMIT – Set Packet Pacing Rate Limit.	2199
29.36.2 QUERY_PP_RATE_LIMIT – Query Packet Pacing Rate Limit	2201
29.36.3 CREATE_SCHEDULING_ELEMENT - Create a Transmit Scheduling element 2202	
29.36.4 DESTROY_SCHEDULING_ELEMENT - Destroy a Transmit Scheduling Element 2207	
29.36.5 QUERY_SCHEDULING_ELEMENT - Query a Transmit Scheduling Element 2208	
29.36.6 MODIFY_SCHEDULING_ELEMENT - Modify a Transmit Scheduling Element Attributes 2210	
29.36.7 CREATE_QOS_PARA_VPORT - Create an E-Switch Scheduler Para-VPort 2212	
29.36.8 DESTROY_QOS_PARA_VPORT - Destroy an E-Switch SchedulerPara-VPort 2213	
29.37 Multicast Groups Commands.	2214
29.37.1 ATTACH_TO_MCG - Attach a QP to Multicast Group	2214
29.37.2 DETACH_FROM_MCG - Detach a QP from multicast group	2216
29.38 IB Port sniffer	2217
29.38.1 ATTACH_TO_SNIFFER - Attach a QP to Port Sniffer.	2217
29.38.2 DETACH_FROM_SNIFFER - Detach a QP from a Port Sniffer	2219
29.39 GET_DROPPED_PACKET_LOG	2220
29.40 Diagnostic Counters Commands	2221
29.40.1 SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters	2222
29.40.2 QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters	2224
29.40.3 QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters.	2226
29.41 Congestion Control Commands.	2228
29.41.1 QUERY_CONG_STATUS - Query Congestion Status.	2228
29.41.2 MODIFY_CONG_STATUS - Query Congestion Status.	2230
29.41.3 QUERY_CONG_PARAMS - Query Congestion parameters	2232
29.41.4 MODIFY_CONG_PARAMS - Modify Congestion Parameters	2243
29.41.5 QUERY_CONG_STATISTICS - Query Congestion Statistics.	2247
29.42 VXLAN Commands	2251
29.42.1 ADD_VXLAN_UDP_DPORT Command	2251
29.42.2 DELETE_VXLAN_UDP_DPORT Command	2252
29.43 Miscellaneous Commands	2253
29.43.1 NOP Command	2253

29.44 EMC Specific Commands	2254
29.44.1 QUERY_RSV_RESOURCES	2254
29.44.2 [EMC] CREATE_MTT - Create MTT Entries	2256
29.44.3 [EMC] DESTROY_MTT - Destroy MTT Entries	2258
29.44.4 [EMC] QUERY_MTT - Query MTT Entries	2259
29.44.5 [EMC] ACCESS_CR	2260
29.44.6 [EMC] MODIFY_SCHED_QUEUE	2261
29.44.7 ARM_RESOURCE_DUMP_EVENT - ARM Resource Dump Event.	2265

Chapter 30 ICMD Command Reference 2267

30.1 Introduction	2267
30.2 Conventions	2267
30.3 Command Summary	2267
30.4 Miscellaneous Commands (MISC)	2273
30.4.1 ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities	2273
30.4.2 ICMD_ACCESS_REGISTER - Register Access	2276
30.4.3 ICMD_MH_SYNC - Multi-Host Synchronization	2280
30.4.4 ICMD_MH_SYNC_STATUS - Multi-Host Synchronization Status	2283
30.4.5 ICMD_QUERY_FW_INFO - Get device firmware information	2283
30.5 Configuration Commands (CONFIG)	2285
30.5.1 ICMD_QUERY_VIRTUAL_MAC - Get Device MAC Address	2285
30.5.2 ICMD_SET_VIRTUAL_MAC - Set Device MAC Address	2287
30.5.3 ICMD_SET_WOL_ROL - Set Wake/Reset-On-LAN	2288
30.5.4 ICMD_QUERY_WOL_ROL - Get Wake/Reset-On-LAN	2289
30.5.5 ICMD_QUERY_VIRTUAL_PARAMETERS - Get Device Virtual Parameters	2290
30.5.6 ICMD_SET_VIRTUAL_PARAMETERS - Set Device Virtual Parameters .	2291
30.6 Debug Commands	2292
30.6.1 ICMD_SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters	2292
30.6.2 ICMD_QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters	2293
30.6.3 ICMD_QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters	2294
30.7 HP Commands (HP)	2295
30.7.1 ICMD_INIT_OCSD	2295
30.7.2 ICMD_OCBB_INIT - Initialize OCBB report	2295
30.7.3 ICMD_OCBB_QUERY_HEADER_STATUS - Get OCBB header initialization status	2296
30.7.4 ICMD_OCBB_QUERY_ETOC_STATUS - Get OCBB ETOC initialization status	2297
30.7.5 ICMD_OCBB_SET_EVENT - Set OCBB event information	2297
30.8 Internal Commands	2298
30.8.1 ICMD_GET_FRC_CHALLENGE - Get Factory Re-customization Token .	2298
30.8.2 ICMD_SET_FRC - Set Factory Re-customization	2301

Revision History

Revision 0.53

Draft A:

1. Added initializing_state to INIT SEGMENT. See Table 14, “Initialization Segment,” on page 234.
2. Renamed resources_on_emulation_manager to resources_on_nvme_emulation_manager in HCA - CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
3. Added resources_on_virtio_net_emulation_manager and resources_on_virtio_blk_emulation_manager to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
4. Added resources_on_emulation_manager to Virtio Net Device Emulation Object. See Table 310, “VIRTIO NET DEVICE EMULATION Object Layout,” on page 582.
5. Added resources_on_emulation_manager to Virtio BLK Emulation object. See Table 318, “VIRTIO BLK DEVICE EMULATION Object Layout,” on page 587.
6. Added NVMe-oF Report Back section. Section 20.5.1, “NVMe-OF Report Back,” on page 1001.
7. Added nvme_stats_report_back, min_iorsz and max_iorsz to NVMeoF Capabilities. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.
8. Added nvme_stats_report_back_en, iorsz, num_bits_for_in_flight_data, report_back_location, report_in_flight_level0, log_report_in_flight_step_size, report_pending_cmds_level0 and log_report_pending_cmds_step_size to NVMe-oF Target Offload Context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
9. Added PAUSE_TX and DEVICE_FLUSH bits to type filed in STRS_STOP_TOGGLE. See Table 904, “STRS_STOP_TOGGLE Register Fields,” on page 1114.
10. Added PAUSE_TX Stop Toggle Modifier. See Table 919, “PAUSE_TX Stop Toggle Modifier Layout,” on page 1121.
11. Added vlan_and_fwd_to_table to Flow Table Properties Capabilities. See Table 1587, “Flow Table Properties Layout,” on page 1753.
12. Added IPsec Lifetime section. See Section 21.7.9, “IPsec Lifetime,” on page 1027.
13. Updated icv_length in IPsec Offload Object. See Table 343, “IPSEC_OFFLOAD Object Fields,” on page 607.
14. Added ece to HCA_CPA. See Table 1571, “HCA Capabilities Layout,” on page 1671.
15. Added ece to QP Commands. See Section 29.10, “QP Commands,” on page 1897.
16. Added 5_SGMT_TYPE_QUERY_CQ, MLX5_SGMT_TYPE_QUERY_MKEY and MLX-5_SGMT_TYPE_QUERY_QP to Pre-defined Resource Segments. See Table 827, “Pre-defined Resource Segments,” on page 1065.
17. Added db_umem_free to REGEXP_PARAMS. See Table 1974, “REGEXP_PARAMS Field Descriptions,” on page 1944.
18. Performed several changes in PDDR register. See Table 1169, “PDDR - Port Diagnostics Database Register Layout,” on page 1362.
19. PPLR: Added phy_remote loopback option and exposed port_type field in Adabe. See Table 1071, “PPLR - Port Physical Loopback Register Fields,” on page 1267.
20. PTYS: Added force_tx_aba_param filed and exposed port_type field in Adabe.

Draft B:

21. [Resized DCT context. See Table 25, “DCT Context Layout,” on page 279.](#)
22. [Added ece to DCT Context. See Table 25, “DCT Context Layout,” on page 279.](#)
23. [Added ooo_enabled and sr_enable to DCT Context. See Table 25, “DCT Context Layout,” on page 279.](#)
24. [Added ece to CREATE_DCT output Command. See Table 2456, “CREATE_DCT Output Structure Layout,” on page 2144.](#)
25. [Added stat_rate_200 to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
26. [Updated stat_rate in Address Path. See Table 22, “Address Path Structure Fields,” on page 274.](#)
27. [Updated static_rate in UD Address Vector. See Table 110, “UD Address Vector Field Descriptions,” on page 362.](#)
28. [Added status_bit to modify_field_select field in VIRTIO_NET Device Emulation Object. See Table 311, “VIRTIO NET DEVICE EMULATION Object Fields,” on page 583.](#)
29. [Added Flow Sampler Object. See Section 8.27.12, “Flow Sampler Object,” on page 560.](#)
30. [Added FLOW_SAMPLER bit to destination_type field in Destination Entry Format. See Table 2159, “Destination Entry Format Field Descriptions,” on page 2032.](#)
31. [Added FLOW_SAMPLER bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
32. [Added log_max_flow_sampler_num to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.](#)
33. [Renamed tism_or_qpn_vhca_id to vhca_id VIRTIO_NET_Q. See Table 304, “VIRTIO_NET_Q Object Format,” on page 573.](#)
34. [Added relaxed_ordering_write_umr and relaxed_ordering_read_umr to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
35. [Added regexp_alloc_onbehalf_umem to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
36. [Added CRYPTO_COMMISSIONING Register. See Section 28.16, “CRYPTO_COMMISSIONING Register,” on page 1136.](#)
37. [Added IMPORT_KEK_HANDLE Register. See Section 28.17, “IMPORT_KEK_HANDLE Register,” on page 1136.](#)
38. [Added CREDENTIAL_HANDLE Register. See Section 28.18, “CREDENTIAL_HANDLE Register,” on page 1138.](#)
39. [Added Device_Buffer_Update event to general notification event. See Table 215, “General Notification Event Subtype,” on page 496.](#)
40. [Added factory_port_guid and factory_node_guid to ICMD_QUERY_VIRTUAL_MAC. See Table 2754, “ICMD_QUERY_VIRTUAL_MAC Output Mailbox Layout,” on page 2285.](#)
41. [Added ooo_enabled and sr_enable to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.](#)
42. [Added sr_enable to INIT2RTR_QP Optional Parameters command. See Table 1905, “INIT2RTR_QP Optional Parameters Mask,” on page 1913.](#)
43. [Added selective_repeat to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
44. [Added log_max_dci_stream_channels and log_max_dci_errored_streams to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
45. [Added log_num_dci_stream_channels and log_num_dci_errored_streams to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.](#)
46. [Added log_num_dci_stream_channels and log_num_dci_errored_streams to CEATE_QP Required Parameters. See Table 1885, “CREATE_QP Required Parameters,” on page 1902.](#)

47. Added `dc_i_stream_channel_id` to `RTS2RTS_QP`. See Table 1912, “RTS2RTS_QP Input Structure Layout,” on page 1918.
48. Added `STREAM_CHANNEL_ACCESS` bit to `op_mod` field in `RTS2RTS_QP`. See Table 1912, “RTS2RTS_QP Input Structure Layout,” on page 1918.
49. Added `dc_i_stream_channel_id` to `WQE Ctrl Segment`. See Table 42, “General - Ctrl Segment Format,” on page 312.
50. Added `num_nv_import_keks` and `num_nv_credentials` to `Crypto Capabilities`. See Table 1625, “Crypto Capabilities Layout,” on page 1802.
51. Changed `NV_HPCC_INT` TLV index.
52. Added `cache_disable` to `NV_HOST_CHAINING_CONF`. See Table 386, “NV_HOST_CHAINING_CONF Layout,” on page 661.
53. Added `cache_disable_supported` to `NV_HOST_CHAINING_CAP`. See Table 388, “NV_HOST_CHAINING_CAP Layout,” on page 662.
54. Added `vector_calc_disable` to `NV_SW_OFFLOAD_CONFIG`. See Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.
55. Added `vector_calc_disable_supported` to `NV_SW_OFFLOAD_CAP`. See Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707.
56. Added `unknown_uplink_mac_flood` to `NV_MPFS_CONF`. See Table 558, “NV_MPFS_CONF Layout,” on page 760.
57. Added `unknown_uplink_mac_flood_supported` to `NV_MPFS_CAP`. See Table 560, “NV_MPFS_CAP Layout,” on page 762.
58. [Internal] Added the following internal reserved TLV enumerations across the `NvConfig` chapter: `NV_GLOBAL_FW_INT_0`, `NV_HOST_FW_INT_0`, `NV_PORT_FW_INT_0`, `NV_PF_FW_INT_0` and `NV_PF_FW_INT_1`.
59. Added `log_max_outstanding_wqe` to `NV_SW_OFFLOAD_CONFIG`. See Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.
60. Added `max_outstanding_wqe_supported` to `NV_SW_OFFLOAD_CAP`. See Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707.
61. Added `np_rtt_int_data_mode`, `np_rtt_response_dscp_valid`, `np_rtt_response_dscp` and `system_rtt` to `NV_HPCC_INT`. See Table 456, “NV_HPCC_INT Layout,” on page 713.
62. Added `NV_HPCC_INT_CAP` section. See Section 11.7.42, “NV_HPCC_INT_CAP,” on page 715.
63. Added `mlxcfname` for `user_cc_en` in `NV_PROG_CC`. See Table 455, “NV_PROG_CC Field Descriptions,” on page 713.
64. Added `exp_rom_nvme_uefi_x86_supported`, `exp_rom_nvme_uefi_arm_supported`, `exp_rom_virtio_blk_uefi_x86_supported`, `exp_rom_virtio_blk_uefi_arm_supported`, `exp_rom_virtio_net_uefi_x86_supported`, `exp_rom_virtio_net_uefi_arm_supported` and `exp_rom_virtio_net_pxe_supported` to `NV_PCI_CAP`. See Table 406, “NV_PCI_CAP Layout,” on page 670.
65. Added `exp_rom_nvme_uefi_x86_enable`, `exp_rom_nvme_uefi_arm_enable`, `exp_rom_virtio_blk_uefi_x86_enable`, `exp_rom_virtio_blk_uefi_arm_enable`, `exp_rom_virtio_net_uefi_x86_enable`, `exp_rom_virtio_net_uefi_arm_enable` and `exp_rom_virtio_net_pxe_enable` to `NV_PCI_CONF`. See Table 404, “NV_PCI_CONF Layout,” on page 668.
66. Added `SLRED` Register. See Section 28.19.49, “SLRED - SerDes Lane Receive Eye Diagram,” on page 1321.

Revision 0.52 (March 2020)

Draft A:

67. Added `event_supported` to Menu Record. See [Table 842, “Menu Record Layout,”](#) on page 1074.
68. Removed `arm_all` from `ARM_RESOURCE_DUMP_EVENT`. See [Table 2727, “ARM_RESOURCE_DUMP_EVENT Input Structure Layout,”](#) on page 2265.
69. Added `sw_owner_reformat_supported` to Flow table NIC Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.
70. Added `packet_pacing_uid` to `HCA_CAP`. See [Table 1597, “Quality of Service Capabilities Layout,”](#) on page 1773.
71. Added `nvmeotcp_en` to TIR Context. See [Table 116, “TIR Context Format,”](#) on page 374.
72. Added `nvmeotcp_klm_list_table_umem_id` to TIR Context. See [Table 116, “TIR Context Format,”](#) on page 374.
73. Added `nvmeotcp` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
74. Added NVMeoTCP Capabilities. See [Table 1623, “NVMeoTCP Capabilities Layout,”](#) on page 1801.
75. Renamed `encryption_standard` to `acc_type` in Transport Static Params. See [Table 260, “Transport Static Params Context Layout,”](#) on page 526.
76. Added `pda`, `hddgst_en`, `ddgst_en`, `hdgst_offload_en`, `ddgst_offload_en`, `zero_copy_en`, `ti` and `nvme_resync_tcp_sn` to Transport Static Params. See [Table 260, “Transport Static Params Context Layout,”](#) on page 526.
77. Added NVMeoTCP Progress Params Context. See [Table 264, “NVMeoTCP Progress Params Context Layout,”](#) on page 529.
78. Added NVMeoTCP Offload section. See [Section 4.10, “NVMeoTCP Offload,”](#) on page 220.
79. Added `nvmeotcp_resync`, `nvmeotcp_data_digest` and `nvmeotcp_zc` to CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
80. Added NVMeoTCP Extra CQE Info. See [Table 148, “NVMeoTCP Extra CQE Info Layout,”](#) on page 448.
81. Added `NVMEOTCP_TIS_PROGRESS_PARAMS` and `NVMEOTCP_TIR_PROGRESS_PARAMS` to `GET_PSV` and `SET_PSV` in `opc_mod` table. See [Table 44, “OPC_MOD Field Encoding,”](#) on page 316.
82. Renamed `flow_meter` to `flow_meter_old` in QoS capabilities. See [Table 1597, “Quality of Service Capabilities Layout,”](#) on page 1773.
83. Added `flow_meter` to QoS Capabilities. See [Table 1597, “Quality of Service Capabilities Layout,”](#) on page 1773.
84. Removed `error_type` from `VIRTIO_NET_Q`. See [Table 304, “VIRTIO_NET_Q Object Format,”](#) on page 573.
85. Removed `error_type` from `VIRTIO_BLK_Q`. See [Table 322, “VIRTIO_BLK_Q Object Format,”](#) on page 593.
86. Added `error_type` to `VIRTIO_Q`. See [Table 306, “VIRTIO_Q Format,”](#) on page 575.
87. Added `flow_table_update` to SQ Context. See [Table 138, “SQ Context Format,”](#) on page 404.
88. Added `wqe_based_flow_table_update_dest_type_offset` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
89. Added `wqe_based_flow_update_action` to Flow Table Properties. See [Table 1588, “Flow Table Properties Field Descriptions,”](#) on page 1754.

90. Added `access_flow_lookup_mode`, `update_flow_lookup_mode`, `log2_match_table_size`, `reparse_log2_match_table_rows`, `flow_match_definer_id` and `flow_hash_definer_id` to `CREATE_FLOW_GROUP`. See Table 2120, “`CREATE_FLOW_GROUP` Input Structure Layout,” on page 2000.

Draft B:

91. Replaced `tls` with `must_be_0` in `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
92. Added `tls_tx` and `tls_rx` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
93. Added `pci_sync_for_fw_update_event` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
94. Added `PCI_SYNC_FOR_FW_UPDATE` event to general notification event. See Table 215, “General Notification Event Subtype,” on page 496.
95. Added `VIRTIO_Q_COUNTERS` object. See Table 308, “`VIRTIO_Q_COUNTERS` Object Format,” on page 580.
96. Added `pci_sync_for_fw_update_start` and `pci_sync_for_fw_update_resp` to MFRL register. See Table 1549, “MFRL - Management Firmware Reset Level Register Fields,” on page 1646.
97. Updated description in the introduction of Resource Dump section. See Section 27.8, “Resource Dump,” on page 1062.
98. Added Pre-defined Resource Segment section. See Section 27.8.1.2.1.1, “Pre-defined Resource Segments,” on page 1065.
99. Updated `num_of_obj1` and `num_of_obj2` description across the document.
100. PMTM: Added the missing opcodes.
101. PPCNT: Group 0x16 - added lanes 4-7.
102. PPCNT: New group for `usr` - added `fc_uncorrtables`.
103. PDDR: Added new error information 43-46 (troubleshooting - both internal and external).
104. PDDR: Link-up info new indications for fast link-up feature.
105. PDDR: `core_to_phy_link_proto_enabled` - exposed HDR and NDR rates.
106. PPRT: Added support for lock status of 8 lanes. PMAOS: Added operational notification.

Draft C:

107. Added `termination_table_raw_traffic` to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
108. Added `NV_HPCC_INT`. See Table 456, “`NV_HPCC_INT` Layout,” on page 713.
109. Added `NV_PROG_CC`. See Table 454, “`NV_PROG_CC` Layout,” on page 712.
110. Added `max_acc_outstanding_read_byte_supported` to `GLOBAL_PCI_CAP`. See Table 432, “`NV_GLOBAL_PCI_CAP` Layout,” on page 686.
111. Added `max_acc_outstanding_read_byte` to `GLOBAL_PCI_CONF_2`. See Table 436, “`NV_GLOBAL_PCI_CONF_2` Layout,” on page 694.
112. Added `auto_power_save_link_down` to `NV_KEEP_LINK_UP`. See Table 496, “`NV_KEEP_LINK_UP` Layout,” on page 727.
113. Added Disk Encryption section. See Section 20.6, “Disk Encryption,” on page 1003.
114. Added `aes_xts` and `crypto` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
115. Added Crypto Capabilities. See Table 1625, “Crypto Capabilities Layout,” on page 1802.

116. Renamed Encryption Object to DEK Object. See Table 294, “DEK Object Format,” on page 564.
117. Added state, has_keytag and opaque to DEK Object. See Table 294, “DEK Object Format,” on page 564.
118. Renamed key_type to key_purpose in DEK Object. See Table 294, “DEK Object Format,” on page 564.
119. Resized key in DEK Object. See Table 294, “DEK Object Format,” on page 564.
120. Added DEK Key Field Construction Summary. See Table 296, “DEK Key Field Construction Summary,” on page 565.
121. Added IMPORT_KEY Object. See Table 297, “IMPORT_KEY Object Format,” on page 567.
122. Added IMPORT_KEY bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
123. Added CREDENTIAL Object. See Section 8.27.16, “CREDENTIAL Object,” on page 568.
124. Added CREDENTIAL Object bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
125. Added crypto_en to Mkey Context. See Table 362, “MKey Context Format,” on page 629.
126. Added bsf_p_type to BSF Context. See Table 695, “Byte Stream Format (BSF) Layout,” on page 902.
127. Updated description of bsf_size in BSF Context. See Table 695, “Byte Stream Format (BSF) Layout,” on page 902.
128. Added Crypto BSF. See Table 697, “Crypto Byte Stream Format (BSF) Layout,” on page 906.
129. Added CRYPTO_LOGIN Object. See Section , “Destroying the object means logout,” on page 570.
130. Added CRYPTO_LOGIN Object bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
131. Added CRYPTO_OPERATIONAL register. See Section 28.15, “CRYPTO_OPERATIONAL Register,” on page 1135.

Draft D:

132. Added init2_lag_tx_port_affinity to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
133. Added lag_tx_port_affinity to INIT2INIT_QP optional params. See Table 1899, “INIT2INIT_QP Optional Parameters Mask,” on page 1909.
134. Added lag_tx_port_affinity to INIT2RTR_QP optional params. See Table 1905, “INIT2RTR_QP Optional Parameters Mask,” on page 1913.
135. Added hairpin_ib_raw, hairpin_eth2ipoib and hairpin_ipoib2eth to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
136. Added hairpin_offload_type to RQ Context. See Table 124, “RQ Context Format,” on page 385.
137. Added hairpin_offload_type to SQ Context. See Table 138, “SQ Context Format,” on page 404.
138. Added LRH_SLID, LRH_DLID, GRH_FL, GRH_TClass, GRH_SGID_127_96, GRH_SGID_95_64, GRH_SGID_63_32, GRH_SGID_31_0, GRH_DGID_127_96, GRH_DGID_95_64, GRH_DGID_63_32, GRH_DGID_31_0, BTH_PKey and BTH_DQPN bits to field field in SET_ACTION. See Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051.

139. Added `outer_lrh_id`, `outer_grh_flow_label`, `outer_grh_tclass`, `outer_grh_gid` and `outer_bth_pkey` to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761](#).
140. Added Special Flow Tables. See [Section 8.18.2.11, “Special Flow Tables,” on page 419](#).
141. Added Hash Split Group section. See [Section 8.18.3.4.1, “Hash Split Group,” on page 422](#).
142. Added Hash Split Flow section. See [Section 8.18.3.9.3, “Hash Split Flow,” on page 426](#).
143. Updated table format of Match Definer Foramt_63. See [Table 348, “Match_Definer_Format_63 Layout,” on page 612](#)
144. Renamed `nvmeotcp_klm_list_table_umem_id` to `nvmeotcp_klm_list_table_mkey` in TIR Context. See [Table 116, “TIR Context Format,” on page 374](#).
145. Added `pd` to `VIRTIO_Q`. See [Table 306, “VIRTIO_Q Format,” on page 575](#).
146. Removed `access_flow_lookup_mode`, `update_flow_lookup_mode`, `flow_match_definer_id` and `flow_hash_definer_id` from `CREATE_FLOW_GROUP`. See [Table 2120, “CREATE_FLOW_GROUP Input Structure Layout,” on page 2000](#).
147. Added `group_type` and `match_definer_id` to `CREATE_FLOW_GROUP`. See [Table 2120, “CREATE_FLOW_GROUP Input Structure Layout,” on page 2000](#).
148. Removed `group_format_hash_table` and `flow_update_access_hash` from `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
149. Added `flow_group_type_hash_split` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
150. Added `release_all_pages` to `HCA_CA`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
151. Added `release_all_pages` to Pages Request Event. See [Table 234, “Pages Request Event Layout,” on page 502](#).
152. Added `NV_GLOBAL_PCI_CAP_3`. See [Section 11.7.30, “NV_GLOBAL_PCI_CAP_3,” on page 541](#).
153. Added `NV_GLOBAL_PCI_CONF_3`. See [Section 11.7.32, “NV_GLOBAL_PCI_CONF_3,” on page 545](#).
154. Reduced `NV_GLOBAL_PCI_CONF_2` to 4B. See [Section 11.7.31, “NV_GLOBAL_PCI_CONF_2,” on page 544](#).
155. Added `cap_total_descriptors` to SBCAM register. See [Table 1324, “SBCAM - Shared Buffer Capabilities Mask Register Layout,” on page 1490](#).
156. Renamed table `RXP_HANG Stop Toggle Modifier` to `RXP_HOST_HANG Stop Toggle Modifier`. See [Table 909, “RXB_HOST_HANG Stop Toggle Modifier Layout,” on page 1118](#).
157. PMTM - new module width field
158. PDDR - new troubleshooting infos (1043-1046)
159. PDDR - modified some width/speed fields to union structure.
160. [Internal] PDDR- added counters for KR timeouts in phy info
161. PPTT - added TX PRBS polarity configuration support
162. PPRT - added RX PRBS polarity configuration support
163. PPCNT - added post FEC BER - group 0x16

- 164.PPCNT - added max retry window in group 0x22
- 165.PCAM - bits 30 (PLR max retries), Bit 31 for post FEC BER and Bit 42- PRBS polarity
- 166.SLTP - clarification about tap values ([internal] - removal of tx_mode).
- 167.**[Internal] PPLM - added policy capability mask**
- [Internal] Copied from older revision history to be exposed for external release**
- 168.Added ipsec_obj_id bit to modify_enable_mask field in SET_FLOW_TABLE_ENTRY. See [Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023.](#)
- 169.Added IPSEC_SYNDROME bit to *field* field in SET_ACTION. See [Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051.](#)
- 170.Added vhca_id_valid to Resource_dump register. See [Table 937, “RESOURCE_DUMP Register Layout,” on page 1130.](#)
- 171.Added rmpn_xrqn to RTS2RTS_QP Optional Parameters Mask. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.](#)
- 172.Added relaxed_ordering_read to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 173.Added relaxed_ordering_read to Mkey Context. See [Table 362, “MKey Context Format,” on page 629.](#)
- 174.Updated description for fl field in Address Path of QP. See [Table 22, “Address Path Structure Fields,” on page 274.](#)
- 175.Added flex_gtp_u_enabled bit to flex_parser_protocols field in HCA_CAP. See [Table 1572, “HCA Capabilities Field Descriptions,” on page 1679.](#)
- 176.Updated the description of ctrl_general_id field in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,” on page 313.](#)
- 177.Renamed TLS Static Params to Transport Static Params. See [Section 8.22, “Transport Static Params Context,” on page 525.](#)
- 178.Added log_min_resource_dump_eq to Debug Capabilities. See [Table 1599, “Debug Capabilities Layout,” on page 1777.](#)
- 179.Added Resource_Dump_Event. See [Section 8.20.11.3, “Resource Dump Event,” on page 489.](#)
- 180.Added ARM_RESOURCE_DUMP_EVENT. See [Section 29.44.7, “ARM_RESOURCE_DUMP_EVENT - ARM Resource Dump Event,” on page 2265.](#)
- 181.Added wq_umem_offset, wq_umem_id and wq_umem_valid to QUERY_QP command. See [Table 1932, “QUERY_QP Output Structure Layout,” on page 1928.](#)
- 182.Added cq_umem_offset, cq_umem_id and cq_umem_valid to QUERY_CQ command. See [Table 1870, “QUERY_CQ Output Structure Layout,” on page 1892.](#)
- 183.Added xrc_srq_umem_offset, xrc_srq_umem_id and xrc_srq_umem_valid to QUERY_XRC_SRQ command. See [Table 2376, “QUERY_XRC_SRQ Output Structure Layout,” on page 2114.](#)
- 184.Added ats to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 185.Added ATS bit to ma_translation_mode field in Mkey Context. See [Table 363, “MKey Context Fields,” on page 631.](#)
- 186.Added ats to UMEM Context. See [Table 2323, “UMEM Context Format,” on page 2092.](#)
- 187.Added Congestion Control Registers chapter. See [Section 28.20, “Congestion Control Registers,” on page 1421.](#)

188. Added Buffer Management Registers chapter. See [Section 28.21, “Buffer Management Registers,” on page 1450.](#)
189. Added sbcam_reg and cwcam_reg to ICMD_QUER_CAPABILITY. See [Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274.](#)
190. Added sbcam_reg and cwcam_reg to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
191. Added resource_dump to Debug capabilities. See [Table 1600, “Debug Capabilities Field Descriptions,” on page 1778.](#)
192. Added resource_dump register. See [Table 937, “RESOURCE_DUMP Register Layout,” on page 1130.](#)
193. Added pnat to CWTP. See [Table 1236, “CWTP - Congestion WRED ECN TClass Profile Register Layout,” on page 1425.](#)
194. Added pnat to CWTPM. See [Table 1240, “CWTPM - Congestion WRED ECN TClass and Pool Register Layout,” on page 1428.](#)
195. Updated OPC_MOD of UMR, SET_PSV and GET_PSV. See [Table 44, “OPC_MOD Field Encoding,” on page 316.](#)
196. Removed valid and pd from TLS Progress Context. See [Table 262, “TLS Progress Params Context Layout,” on page 528.](#)
197. Updated SET_PSV description to support TLS Progress Context. See [Table 58, “SET_PSV Segment Format,” on page 327.](#)
198. Added ipsec_offload to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
199. Added IPSEC Capabilities. See [Table 29.3.4.15, “IPSEC Capabilities,” on page 1789.](#)
200. Added insert_trailer to Per Protocol Networking Offload Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.](#)
201. Added trailer_alignment, trailer_header_association and insert_trailer to ETH Segment. See [Table 47, “Eth Segment Format,” on page 318.](#)
202. Added ipsec_syndrome and ipsec_next_header to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761.](#)
203. Added ipsec_syndrome and ipsec_next_header to Flow Table Entry Match Set Misc Parameters 2. See [Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013.](#)
204. Added IPSEC_OFFLOAD bit to *general_obj_types* field in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
205. Added IPSEC Offload Object. See [Section 8.27.27, “IPSEC Offload Object,” on page 606.](#)
206. Added ACCESS_ASO bit to Opcode field in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,” on page 313.](#)
207. Added ACCESS_ASO to OPC_MOD Table. See [Table 44, “OPC_MOD Field Encoding,” on page 316.](#)
208. Added ACCESS_ASO to WQE Construction Summary. See [Table 106, “WQE Construction Summary,” on page 355.](#)
209. Added ASO CTRL Segment. See [Section 8.9.4.1.20, “ASO CTRL Segment,” on page 346.](#)
210. Added ASO Data Segment. See [Section 8.9.4.1.21, “ASO Data Segment,” on page 350.](#)

211. Added `ADD_ESP_TRANSPORT`, `L2_TO_L3_ESP_TUNNEL`, `ADD_ESP_TRANSPORT_OVER_UDP`, `DEL_ESP_TRANSPORT`, `L3_ESP_TUNNEL_TO_L2` and `DEL_ESP_TRANSPORT_OVER_UDP` bits to `reformat_type` field in Packet Reformat Context. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,”](#) on page 2045.
212. Added `reformat_add_esp_transport`, `reformat_l2_to_l3_esp_tunnel`, `reformat_add_esp_transport_over_udp`, `reformat_del_esp_transport`, `reformat_l3_esp_tunnel_to_l2` and `reformat_del_esp_transport_over_udp` to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
213. Added `log_max_ipsec_offload` to IPSEC Capabilities. See [Table 1611, “IPSEC Capabilities Layout,”](#) on page 1790.
214. Added `IPSEC_OFFLOAD` bit to action field in Flow Context. See [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.
215. Added `ipsec_obj_id` to Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,”](#) on page 2025.
216. Added `ipsec_encrypt` and `ipsec_decrypt` to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
217. Added IPSEC offload section. See [Section 21.7, “IPsec Offload,”](#) on page 1022.
218. Added Resource Dump section. See [Section 27.8, “Resource Dump,”](#) on page 1062.
219. Added `log_max_uctx` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
220. Added bit `FLOW_METER` to *general_obj_types* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
221. Added `flex_parser_id_icmp_dw1`, `flex_parser_id_icmp_dw0`, `flex_parser_id_icmpv6_dw1`, `flex_parser_id_icmpv6_dw0`, `flex_parser_id_outer_first_mpls_over_gre` and `flex_parser_id_outer_first_mpls_over_udp_label` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
222. Added bits `flex_icmp_enabled` and `flex_icmpv6_enabled` to *flex_parser_protocols* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
223. Added bits `OUT_TCP_ACK_NUM`, `IN_TCP_ACK_NUM`, `OUT_TCP_SEQ_NUM` and `IN_TCP_SEQ_NUM` to *field* field in `SET_ACTION`. See [Table 2205, “SET_ACTION Input Structure Field Descriptions,”](#) on page 2051.
224. Added Flow Meter Object. See [Section 8.27.11, “Flow Meter Object,”](#) on page 556.
225. Added support for Shared Q Counters in `ALLOC_Q_COUNTER` command. See [Table 2553, “ALLOC_Q_COUNTER Input Structure Field Descriptions,”](#) on page 2186.
226. Added `esw_fdb_ipv4_ttl_modify` to Flow Table E-switch Capabilities Layout. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
227. Added `CR_DUMP_TO_MEM` bit to `core_dump_type` field in `CORE_DUMP` Register. See [Table 935, “CORE_DUMP Register Layout,”](#) on page 1128.
228. Added `more_dump`, `mkey`, `address`, `size` and `cookie` to `CORE_DUMP` Register. See [Table 935, “CORE_DUMP Register Layout,”](#) on page 1128.
229. Added `CREATE_UCTX`. See [Section 29.23.5, “CREATE_UCTX,”](#) on page 2087.
230. Added `QUERY_UCTX`. See [Section 29.23.6, “QUERY_UCTX,”](#) on page 2088.
231. Added `DESTROY_UCTX`. See [Section 29.23.7, “DESTROY_UCTX,”](#) on page 2090.
232. Added `CREATE_UMEM`. See [Section 29.23.8, “CREATE_UMEM,”](#) on page 2090.

233. Added QUERY_UMEM. See [Section 29.23.9, “QUERY_UMEM,” on page 2093.](#)
234. Added DESTROY_UMEM. See [Section 29.23.10, “DESTROY_UMEM,” on page 2094.](#)
235. Added Power Failure Event subtype under General Notification Event Subtype table. See [Table 215, “General Notification Event Subtype,” on page 496.](#)
236. Added flex_geneve_tlv_option_0_enabled bit to *flex_parser_protocols* field in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
237. Added flex_parser_id_geneve_tlv_option_0 to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
238. Added max_geneve_tlv_option_data_len and max_geneve_tlv_options to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
239. Added GENEVE_TLV_OPTION bit to *general_obj_types* field in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
240. Added GENEVE TLV OPTION Object. See [Section 8.27.13, “GENEVE TLV OPTION Object,” on page 562.](#)
241. Added DEK Object section. See [Section 8.27.14, “DEK Object,” on page 564.](#)
242. Added geneve_tlv_option_0_data to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761.](#)
243. Added geneve_tlv_option_0_data to Flow Table Entry Match Set Misc parameters 3. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,” on page 2016.](#)
244. Added max_flow_meter_bs_exponent to QoS Capabilities. See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773.](#)
245. Added TLS Offload section. See [Section 4.9, “TLS Offload,” on page 216.](#)
246. Added tls_tx and tls_rx to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
247. Added TLS Capabilities. See [Section 29.3.4.14, “TLS Capabilities,” on page 1788.](#)
248. Added TLS Static Params Context. See [Section 8.22, “Transport Static Params Context,” on page 525.](#)
249. Added TLS Progress Params Context. See [Section 8.23, “TLS PROGRESS Params Context,” on page 528.](#)
250. Added tls_en to TIR Context. See [Table 116, “TIR Context Format,” on page 374.](#)
251. Added tls_en to TIS Context. See [Table 121, “TIS Context Format,” on page 382.](#)
252. Added log_max_dek to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
253. Renamed immediate_or_invalidation_key_or_umr_mkey_or_remote_micro_app_operation_code to ctrl_general_id in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,” on page 313.](#)
254. Added outer_gre_c_present, outer_gre_k_present and outer_gre_s_present to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761.](#)
255. Added wqe_index_ignore_cap to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
256. Added Encapsulation Entropy section. See [Section 8.18.3.7, “Encapsulation Entropy,” on page 423.](#)

257. Added flow_meter_reg_id to QoS Capabilities. See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773.](#)
258. Added sw_steering_vport_icm_address_rx and sw_steering_vport_icm_address_tx to Esw vport Context. See [Table 27, “Esw_vport Context Layout,” on page 284.](#)
259. Added sw_owner_icm_root_0 and sw_owner_icm_root_1 to Flow Table Context. See [Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989.](#)
260. Added sw_steering_fdb_action_drop_icm_address_rx, sw_steering_fdb_action_drop_icm_address_tx, sw_steering_uplink_icm_address_rx and sw_steering_uplink_icm_address_tx to Flow Table Eswitch Capabilities. See [Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748.](#)
261. Added flow_counter_bulk_alloc to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
262. Added flow_counter_bulk to ALLOC_FLOW_COUNTER. See [Table 2174, “ALLOC_FLOW_COUNTER Input Structure Layout,” on page 2038.](#)
263. Renamed QUERY_HOST_PARAMS to QUERY_ESW_FUNCTIONS. See [Section 29.29, “QUERY_ESW_FUNCTIONS - Query e-Switch Functions,” on page 2140.](#)
264. Renamed HOST_PARAMS_CHANGE to ESW_FUNCTIONS_CHANGED. See [Section 8.20.14.4, “ESW_FUNCTIONS_CHANGED Event,” on page 503.](#)
265. Added table_type, destination_table_id, sw_steering_icm_address_rx and sw_steering_icm_address_tx to Flow Meter Object. See [Table 286, “Flow Meter Object Format,” on page 556.](#)
266. Renamed flow_meter_srtcm to flow_meter in QoS Capabilities. See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773.](#)
267. Removed byte_token_ratio from Flow Meter Parameters. See [Table 288, “Flow Meter Parameters Format,” on page 558.](#)
268. Added meter_mode to Flow Meter Parameters. See [Table 288, “Flow Meter Parameters Format,” on page 558.](#)
269. Added uctx_en to NV_SW_OFFLOAD_CONFIG. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.](#)
270. Added uctx_supported to NV_SW_OFFLOAD_CAP. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707.](#)
271. Added pd to TIS context. See [Table 121, “TIS Context Format,” on page 382.](#)
272. Added tls_offload to CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,” on page 435.](#)
273. Added fdb_selection_mode to LAG Context. See [Table 1762, “LAG CONTEXT Structure Layout,” on page 1851.](#)
274. Added Object Change Event. See [Section 8.20.9, “Object Change Event,” on page 481.](#)
275. Added event_cap to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
276. Added Event Capabilities. See [Section 29.3.4.18, “Event Capabilities,” on page 1795.](#)
277. Resized event_bitmask in CREATE_EQ command. See [Table 1843, “CREATE_EQ Input Structure Layout,” on page 1880.](#)

Revision 0.51

Draft A:

278. Added `event_mode`, `full_emulation`, `offload_type` and `device_emulation_id` to `VIRTIO_Q`. See Table 306, “`VIRTIO_Q` Format,” on page 575.
279. Renamed `event_cqn` to `event_cqn_or_msix` in `VIRTIO_Q`. See Table 306, “`VIRTIO_Q` Format,” on page 575.
280. Added `event_mode` to `VIRTIO` Emulation Capabilities. See Table 1615, “`VIRTIO` Emulation Capabilities Layout,” on page 1793.
281. Added Hotplug Device Object. See Section 8.27.22, “HotPlug Device Object,” on page 595.
282. Added `VIRTIO_BLK_Q` Object. See Section 8.27.21, “`VIRTIO_BLK_Q` Object,” on page 591.
283. Added Multi User QP section. See Section 8.3.4.5, “Multi User QP,” on page 256.
284. Added `max_multi_user_group_size` to `HCA_CAP`. See Table 1571, “`HCA` Capabilities Layout,” on page 1671.
285. Added `multi_user_qp_type` and `multi_user_group_size` to QP Context. Table 19, “QP Context Structure Layout,” on page 258.
286. Renamed `remote_qpn` and `remote_qpn_or_multi_user_master_qp` in QP Context. Table 19, “QP Context Structure Layout,” on page 258.
287. Updated the description of `ipsec_syndrome` in IPSEC Syndromes. See Table 807, “IPSEC Syndromes Description,” on page 1030.
288. Added `nic_element_type` and `nic_tsar_type` to QoS Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
289. Removed `crypto_op` from `IPSEC_OFFLOAD` Object. See Table 342, “`IPSEC_OFFLOAD` Object Format,” on page 606.
290. Resized `modify_enable_mask` in `SET_FLOW_TABLE_ENTRY`. See Table 2146, “`SET_FLOW_TABLE_ENTRY` Input Structure Layout,” on page 2023.
291. Added `ipsec_obj_id` bit to `modify_enable_mask` field in `SET_FLOW_TABLE_ENTRY`. See Table 2146, “`SET_FLOW_TABLE_ENTRY` Input Structure Layout,” on page 2023.
292. Removed `IPSEC_OFFLOAD` bit from action field in Flow Context. See Table 2149, “`FLOW_CONTEXT` Field Descriptions,” on page 2026.
293. Added `IPSEC_Decrypt` and `IPSEC_Encrypt` bits to action field in Flow Context. See Table 2149, “`FLOW_CONTEXT` Field Descriptions,” on page 2026.
294. Added `IPSEC_SYNDROME` bit to *field* field in `SET_ACTION`. See Table 2205, “`SET_ACTION` Input Structure Field Descriptions,” on page 2051.
295. Added `outer_ipv6_extension` and `inner_ipv6_extension` to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
296. Added `outer_ipv6_extension` and `inner_ipv6_extension` to Flow Table Entry Match Set Misc Parameters. See Table 2131, “Flow Table Entry Match Set Misc Parameters Fields,” on page 2010.
297. Added Trailer Insertion. See Section 4.3.16, “Trailer Insertion,” on page 206.
298. Added `nic_rx_rdma_fwd_tir` to Flow Table NIC Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
299. Added `RXB_HOST_HANG` bit to type field in `STRS_STOP_TOGGLE` Register. See Table 904, “`STRS_STOP_TOGGLE` Register Fields,” on page 1114.
300. Added `RXP_HANG` Stop Toggle Modifier. See Table 909, “`RXB_HOST_HANG` Stop Toggle Modifier Layout,” on page 1118.

301. Added `sw_owner_v2` to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
 302. Added `steering_format_version` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
 303. Added `header_modify_argument_sw_icm_start_address`, `header_modify_pattern_sw_icm_start_address`, `log_header_modify_argumen_sw_icm_size` and `log_header_modify_pattern_sw_icm_size` to Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
 304. Added `ncfg_reg` to ICMD_QUERY_CAPABILITY. See Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274.
 305. Added NCFG register. See Table 941, “NCFG Register Layout,” on page 1134.
 306. Added LIVE_FW_PATCH subtype event to General Notification Event. See Table 215, “General Notification Event Subtype,” on page 496.
 307. Renamed Device Emulation Object to NVMe Device Emulation Object. See Table 270, “NVMe Device Emulation Object Format,” on page 541.
 308. Added `resources_on_emulation_manager` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
 309. Added `resources_on_emulation_manager` to NVMe Device Emulation Object. See Table 270, “NVMe Device Emulation Object Format,” on page 541.
 310. Added `emulated_device_dma_mkey` to NVMe SQ Object. See Table 272, “NVME_SQ Object Format,” on page 543.
 311. Added `virtio_version_1_0`, `tso_ipv4`, `tso_ipv6`, `tx_csum` and `rx_csum` to Virtio Emulation Capabilities. See Table 1615, “VIRTIO_Emulation Capabilities Layout,” on page 1793.
 312. Added `tso_ipv4`, `tso_ipv6`, `tx_csum` and `rx_csum` to Virtio_Net_Q. See Table 304, “VIRTIO_NET_Q Object Format,” on page 573.
 313. Added `virtio_version_1_0` to VIRTIO_Q. See Table 306, “VIRTIO_Q Format,” on page 575.
 314. Added `gb_valid` to PMDR register. See Table 1038, “PMDR- Port Mapping Definitions Register Layout,” on page 1242.
 315. Added `mcm_tile_valid` to PMDR register. See Table 1038, “PMDR- Port Mapping Definitions Register Layout,” on page 1242.
- Draft B:**
316. Added `vhca_id_valid` to Resource_dump register. See Table 937, “RESOURCE_DUMP Register Layout,” on page 1130.
 317. Added `enabled` and `vhca_id` to VIRTIO NET DEVICE EMULATUION. See Table 310, “VIRTIO NET DEVICE EMULATION Object Layout,” on page 582.
 318. Added `enabled` and `vhca_id` to VIRTIO BLK DEVICE EMULATUION. See Table 318, “VIRTIO BLK DEVICE EMULATION Object Layout,” on page 587.
 319. Added `max_emulated_devices` to VIRTIO Emulation Capabilities. See Table 1615, “VIRTIO_Emulation Capabilities Layout,” on page 1793.
 320. Renamed `total_emulated_pfs` to `max_emulated_devices` in NVME Emulation Capabilities. See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787.
 321. Added `notify_off_multiplier` to VIRTIO Device. See Table 312, “VIRTIO DEVICE Layout,” on page 584.

- 322.Added arm, enabled and device_status bits to modify_field_select field in VIRTIO NET DEVICE EMULATUION. See Table 310, “VIRTIO NET DEVICE EMULATION Object Layout,” on page 582.
- 323.Added arm, enabled and device_status bits to modify_field_select field in VIRTIO BLK DEVICE EMULATUION. See Table 318, “VIRTIO BLK DEVICE EMULATION Object Layout,” on page 587.
- 324.Added device_pci_parameters to HotPlug Device Object. See Table 324, “HotPlug Device Object Format,” on page 595.
- 325.Added device_pci_parameters to VIRTIO Device Object. See Table 312, “VIRTIO DEVICE Layout,” on page 584.
- 326.Added device_pci_parameters to NVME Device Object. See Table 270, “NVMe Device Emulation Object Format,” on page 541.
- 327.Removed vendor_id and subsystem_vendor_id from NVME Device Object. See Table 270, “NVMe Device Emulation Object Format,” on page 541.
- 328.Added match_definer_format_supported and max_num_match_definer to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 329.Added MATCH_DEFINER bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 330.Added Match Definer Object. See Section 8.27.28, “Match Definer Object,” on page 611.

Draft C:

- 331.Added initial_registers_valid to HotPlug Device Object. See Table 324, “HotPlug Device Object Format,” on page 595.
- 332.Added emulation_initial_registers to HotPlug Device Object. See Table 324, “HotPlug Device Object Format,” on page 595.
- 333.Added log_max_frontend_nsid to NVMeoF capabilities. See HCA_CAP. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.
- 334.Removed enabled bit from modify_field_select field in Virtio_Net_Device_Emulation. See Table 310, “VIRTIO NET DEVICE EMULATION Object Layout,” on page 582.
- 335.Removed enabled bit from modify_field_select field in Virtio_BLK_Device_Emulation. See Table 318, “VIRTIO BLK DEVICE EMULATION Object Layout,” on page 587.
- 336.Added qpn_vhca_id to Virtio_Net_Q. See Table 304, “VIRTIO_NET_Q Object Format,” on page 573.
- 337.Added qpn_vhca_id to Virtio_BLK_Q. See Table 322, “VIRTIO_BLK_Q Object Format,” on page 593.
- 338.Added rts2rts_qp_rmp to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 339.Added rmpn_xrqn to RTS2RTS_QP optional parameters Mask. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.
- 340.Added relaxed_ordering_read to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 341.Added relaxed_ordering_read to Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- 342.Added fl_rc_qp_when_roce_disabled to RoCE capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732.

- 343. Updated fl description in Address Path of QP. See Table 22, “Address Path Structure Fields,” on page 274.
- 344. Renamed CQ_MODE to QP_MODE in event type in VIRTIO_Q. See Table 307, “VIRTIO_Q Fields Description,” on page 577.
- Draft D:**
- 345. Added sw_network_metadata to Eswitch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
- 346. Added sw_network_metadata to NIC Vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- 347. Added sw_network_metadata to MODIFY NIC Vport Context Bitmask. See Table 35, “MODIFY_NIC_VPORT_CONTEXT Bitmask,” on page 293.
- 348. Resized intr in EQ Context. See Table 244, “Event Queue Context Layout,” on page 505.
- 349. Added vhca_id to HotPlug Device Object. See Table 324, “HotPlug Device Object Format,” on page 595.
- 350. Added staging_buffer_umem_id, staging_buffer_umem_size and staging_buffer_umem_offset to NVMMF Target Offload Context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- 351. Added NVMMF_BACKEND_CONTROLLER, NVMMF_NAMESPACE and NVMMF_NAMESPACE_COUNTER_SET bits to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 352. Added NVMMF_BACKEND_CONTROLLER Object. See Section 8.27.23, “NVMMF_BACKEND_CONTROLLER Object,” on page 600.
- 353. Added NVMMF_NAMESPACE Object. See Section 8.27.24, “NVMMF_NAMESPACE Object,” on page 602.
- 354. Added NVMMF_NAMESPACE_COUNTER_SET Object. See Section 8.27.25, “NVMMF_NAMESPACE_COUNTER_SET Object,” on page 603.
- 355. Added another command opcodes for QUERY_XRQ_DC_PARAMS_ENTRY, SET_XRQ_DC_PARAMS_ENTRY, QUERY_XRQ_ERROR_PARAMS and RELEASE_XRQ_ERROR commands. See Table 1550, “Commands List Sorted by Opcode,” on page 1647.
- 356. Updated Object Change Event section. See Section 8.20.9, “Object Change Event,” on page 481.
- 357. Added max_tunnel_desc to Virtio Emulation Capabilities. See Table 1615, “VIRTIO_Emulation Capabilities Layout,” on page 1793.
- 358. Added max_tunnel_desc to VIRTIO_Q. See Table 306, “VIRTIO_Q Format,” on page 575.
- 359. Removed arm from VIRTIO NET Device Emulation Object. See Table 310, “VIRTIO NET DEVICE EMULATION Object Layout,” on page 582.
- 360. Removed arm bit from modify_field_select field in VIRTIO NET Device Emulation Object. See Table 311, “VIRTIO NET DEVICE EMULATION Object Fields,” on page 583.
- 361. Removed arm from VIRTIO BLK Device Emulation Object. See Table 318, “VIRTIO BLK DEVICE EMULATION Object Layout,” on page 587.
- 362. Removed arm bit from modify_field_select field in VIRTIO BLK Device Emulation Object. See Table 319, “VIRTIO BLK DEVICE EMULATION Object Fields,” on page 588.
- 363. Added flex_gtp_u_enabled bit to flex_parser_protocols field in HCA_CAP. See Table 1572, “HCA Capabilities Field Descriptions,” on page 1679.

364. Added `gtpu_teid`, `gtpu_msg_type` and `gtpu_msg_flags` to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
365. Added `tunnel_stateless_gtp` to Per Protocol Networking Offload Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
366. Added `gtpu_teid`, `gtpu_msg_type` and `gtpu_msg_flags` to Flow Table Entry Misc 3. See [Table 2135, “Flow Table Entry Match Set Misc Parameters 3 Fields,”](#) on page 2016.
367. Added `FLOW_TABLE_ACCESS` bit to Opcode field in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,”](#) on page 313.
368. Added `FLOW_TABLE_ACCESS` bit to Opcode_MOD. See [Table 44, “OPC_MOD Field Encoding,”](#) on page 316.
369. Updated the description of `ctrl_general_id` field in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,”](#) on page 313.
370. Added Flow Update Control Segment. See [Section 8.9.4.1.22, “Flow Update Control Segment,”](#) on page 351.
371. Added Flow Update Data Segment. See [Section 8.9.4.1.23, “Flow Update Data Segment,”](#) on page 352.
372. Added `wqe_based_flow_table_update`, `group_format_hash_table` and `flow_update_access_hash` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
373. Added WQE Based Flow Table Capabilities. See [Table 1621, “WQE_BASED_FLOW_TABLE Capabilities Layout,”](#) on page 1799.
374. Added `wqe_based_flow_update` to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
375. Added `wqe_based_flow_update` to Flow Table Context. See [Table 2102, “FLOW_TABLE CONTEXT Structure Layout,”](#) on page 1989.
376. Added `tunnel_ecn_copy_offload_disable` to `NV_SW_OFFLOAD_CONFIG`. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,”](#) on page 702.
377. Added `tunnel_ecn_copy_offload_disable_supported` to `NV_SW_OFFLOAD_CAP`. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,”](#) on page 707.
378. Added `flex_parser_profile_3_supported` to `NV_FLEX_PARSER_CAP`. See [Table 400, “NV_FLEX_PARSER_CAP Layout,”](#) on page 666.
379. Added `nvme_emu_revision_id`, `nvme_emu_subsystem_id`, `nvme_emu_subsystem_vendor_id` and `nvme_emu_num_msix` to `NV_EMULATION_NVME_CONF`. See [Table 568, “NV_EMULATION_NVME_CONF Layout,”](#) on page 767.
380. Resized `nvme_emu_class_code` in `NV_EMULATION_NVME_CONF`. See [Table 568, “NV_EMULATION_NVME_CONF Layout,”](#) on page 767.
381. Updated the description of `num_pfs` field in `NV_GLOBAL_PCI_CONF`. See [Table 430, “NV_GLOBAL_PCI_CONF Layout,”](#) on page 682.
382. Added `NV_EMULATION_PCI_SWITCH_CONF`. See [Section 11.7.99, “NV_EMULATION_PCI_SWITCH_CONF,”](#) on page 769.
383. Added `NV_EMULATION_PCI_SWITCH_CAP`. See [Section 11.7.100, “NV_EMULATION_PCI_SWITCH_CAP,”](#) on page 770.

- 384. Added NV_EMULATION_VIRTIO_NET_CONF. See [Section 11.7.101, “NV_EMULATION_VIRTIO_NET_CONF,”](#) on page 770.
- 385. Added NV_EMULATION_VIRTIO_NET_CAP. See [Section 11.7.102, “NV_EMULATION_VIRTIO_NET_CAP,”](#) on page 772.
- 386. Added NV_EMULATION_VIRTIO_BLK_CONF. See [Section 11.7.103, “NV_EMULATION_VIRTIO_BLK_CONF,”](#) on page 773.
- 387. Added NV_EMULATION_VIRTIO_BLK_CAP. See [Section 11.7.104, “NV_EMULATION_VIRTIO_BLK_CAP,”](#) on page 775.
- 388. Added end to Management Debug Session Register (MDSR). See [Table 1544, “MDSR - Management Debug Session Register Layout,”](#) on page 1642.

Draft E:

- 389. Added sig_crc32c and sig_crc64_xp10 to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 390. Added CRC64_xp10 and CRC32c bits to sig_type field in Signature Rules Fields. See [Table 700, “Signature Rules Fields,”](#) on page 909.
- 391. Added CRC64_xp10 and CRC32c to sig_type in Signature Rules Table. See [Table 701, “Signature Rules Table,”](#) on page 910.
- 392. Renamed TLS Static Params to Transport Static Params. See [Section 8.22, “Transport Static Params Context,”](#) on page 525.
- 393. Removed pd from NVME SQ. See [Table 272, “NVME_SQ Object Format,”](#) on page 543.
- 394. Added Device Emulation Chapter. See [Section 20.4, “Device Emulation,”](#) on page 994.
- 395. Added nvmeotcp_en to TIR Context. See [Table 116, “TIR Context Format,”](#) on page 374.
- 396. Added log_min_resource_dump_eq to Debug Capabilities. See [Table 1599, “Debug Capabilities Layout,”](#) on page 1777.
- 397. Added Resource_Dump_Event. See [Section 8.20.11.3, “Resource Dump Event,”](#) on page 489.
- 398. Added ARM_RESOURCE_DUMP_EVENT. See [Section 29.44.7, “ARM_RESOURCE_DUMP_EVENT - ARM Resource Dump Event,”](#) on page 2265.

Revision 0.50

Draft A:

- 399. Added PXE_WO_FAIL_ISCSI and ISCSI_WO_FAIL_PXE to legacy_boot_protocol in NV_ROM_BOOT_CONF1. See [Table 597, “NV_ROM_BOOT_CONF1 Field Descriptions,”](#) on page 788.
- 400. Added legacy_boot_wo_failover_supported to NV_ROM_CAP. See [Table 590, “NV_ROM_CAP Layout,”](#) on page 782.
- 401. Updated key_size in DEK. See [Table 295, “DEK Fields,”](#) on page 564.
- 402. Added esw_fdb_modify_header_fwd_to_table to Flow Table E-switch Capabilities. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
- 403. Added wq_umem_offset, wq_umem_id and wq_umem_valid to QUERY_QP command. See [Table 1932, “QUERY_QP Output Structure Layout,”](#) on page 1928.
- 404. Added cq_umem_offset, cq_umem_id and cq_umem_valid to QUERY_CQ command. See [Table 1870, “QUERY_CQ Output Structure Layout,”](#) on page 1892.
- 405. Added xrc_srq_umem_offset, xrc_srq_umem_id and xrc_srq_umem_valid to QUERY_XRC_SRQ command. See [Table 2376, “QUERY_XRC_SRQ Output Structure Layout,”](#) on page 2114.

- 406. Added `match_uplink_other_esw` to E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- 407. Added VIRTIO Emulation section. See [Section 20.7, “VIRTIO Emulation Offload,”](#) on page 1008.
- 408. Updated VIRTIO_NET_Q Object. See [Table 304, “VIRTIO_NET_Q Object Format,”](#) on page 573.
- 409. Added `RELEASE_XRQ_ERROR` Command. See [Section 29.27.8, “RELEASE_XRQ_ERROR – Release XRQ Error,”](#) on page 2128.
- 410. Added `release_xrq_error` to NVMe-oF Capabilities. See [Table 1603, “NVMe-oF Capabilities Layout,”](#) on page 1780.
- 411. Added `ats` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 412. Added `ATS` bit to `ma_translation_mode` field in Mkey Context. See [Table 363, “MKey Context Fields,”](#) on page 631.
- 413. Added `ats` to UMEM Context. See [Table 2323, “UMEM Context Format,”](#) on page 2092.
- 414. Added `NV_VIRT_GUID_ADDR` TLV. See [Section 11.7.22, “NV_VIRT_GUID_ADDR,”](#) on page 677.
- 415. Added Congestion Control Registers chapter. See [Section 28.20, “Congestion Control Registers,”](#) on page 1421.
- 416. Added Buffer Management Registers chapter. See [Section 28.21, “Buffer Management Registers,”](#) on page 1450.
- 417. Removed `virtual_node_guid_en` from `ICMD_SET_VIRTUAL_MAC` Output. See [Table 2754, “ICMD_QUERY_VIRTUAL_MAC Output Mailbox Layout,”](#) on page 2285.
- 418. Removed `virtual_node_guid_en` from `ICMD_SET_VIRTUAL_MAC` Input. See [Table 2756, “ICMD_SET_VIRTUAL_MAC Input Mailbox Layout,”](#) on page 2287.

Draft B:

- 419. Added `vf_vpd_enable` to `NV_GLOBAL_PCI_CONF`. See [Table 430, “NV_GLOBAL_PCI_CONF Layout,”](#) on page 682.
- 420. Added `vf_vpd_supported` to `NV_GLOBAL_PCI_CAP`. See [Table 432, “NV_GLOBAL_PCI_CAP Layout,”](#) on page 686.
- 421. Added `max_sgl_for_optimized_preformance` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 422. Added `error_report_filter`, `error_report_qpn`, `lossless_error_report` and `never_mask_more_bit` to NVMF Target Offload Context. See [Table 256, “NVMe-oF Target Offload Context Header Layout,”](#) on page 519.
- 423. Added NVMF Error Message. See [Table 258, “NVMF Error Message Layout,”](#) on page 524.
- 424. Added `nvmf_error_message_size`, `nvmf_error_reporting`, `nvmf_error_filters` and `lossless_error_report` to NVMF Capabilities. See [Table 1603, “NVMe-oF Capabilities Layout,”](#) on page 1780.
- 425. Added `tunnel_stateless_geneve_tx`, `tunnel_stateless_mpls_over_udp`, `tunnel_stateless_mpls_over_gre`, `tunnel_stateless_vxlan_gpe`, `tunnel_stateless_ipv4_over_vxlan` and `tunnel_stateless_ip_over_ip` to Per Protocol Networking Offload Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- 426. Added `vnic_env_int_rq_oob` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.

- 427. Added `internal_rq_out_of_buffer` to vNIC Diagnostic Statistics. See [Table 2550, “vNIC Diagnostic Statistics Layout,”](#) on page 2183.
- 428. Added `ingress_acl_flow_counter_for_drop_actions` to Flow Table E-switch Capabilities. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
- 429. Added MMO Section. See [Section 21.6, “MMO - Memory to Memory Offloads,”](#) on page 1018.
- 430. Added REGEXP Section. See [Section 21.6.1, “REGEXP MMO,”](#) on page 1019.
- 431. Added `regexp`, `regexp_num_of_engines` and `regexp_log_crspcsize` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 432. Added MMO bit to Opcode in Ctrl Segment. See [Table 43, “General Ctrl Segment Field,”](#) on page 313.
- 433. Updated `ctrl_general_id` in General Ctrl Segment to indicate `mmo_control`. See [Table 43, “General Ctrl Segment Field,”](#) on page 313.
- 434. Added REGEXP to OPC_MOD Table. See [Table 44, “OPC_MOD Field Encoding,”](#) on page 316.
- 435. Added MMO MetaData Segment. See [Section 8.9.4.1.19, “MMO Metadata Segment,”](#) on page 346.
- 436. Added MMO MetaData Segment column to WQE Construction Summary. See [Table 106, “WQE Construction Summary,”](#) on page 355.
- 437. Added REGEXP commands. See [Section 29.11, “REGEXP Commands,”](#) on page 1942.
- 438. Added `sbcam_reg` and `cwcaml_reg` to ICM-D_QUERY_CAPABILITY. See [Table 2736, “ICM-D_QUERY_CAPABILITY General Capability Output Mailbox Layout,”](#) on page 2274.
- 439. Added `sbcam_reg` and `cwcaml_reg` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 440. Added `ats_enabled` to NV_PCI_CONF. See [Table 404, “NV_PCI_CONF Layout,”](#) on page 668.
- 441. Added `ats_supported` to NV_PCI_CAP. See [Table 406, “NV_PCI_CAP Layout,”](#) on page 670.
- 442. Added `modify_xrq` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 443. Added MODIFY_XRQ command. See [Section 29.27.4, “MODIFY_XRQ – Modify XRQ,”](#) on page 2120.
- 444. Added `resource_dump` to Debug capabilities. See [Table 1600, “Debug Capabilities Field Descriptions,”](#) on page 1778.
- 445. Added `resource_dump` register. See [Table 937, “RESOURCE_DUMP Register Layout,”](#) on page 1130.
- 446. Added `outer_geneve_ver` and `outer_geneve_c` to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- 447. Added `geneve_ver` and `geneve_c` to Flow Table Entry Match Set Misc Parameters. See [Table 2130, “Flow Table Entry Match Set Misc Parameters Format,”](#) on page 2009.
- 448. Added `pnat` to CWTP. See [Table 1236, “CWTP - Congestion WRED ECN TClass Profile Register Layout,”](#) on page 1425.
- 449. Added `pnat` to CWTPM. See [Table 1240, “CWTPM - Congestion WRED ECN TClass and Pool Register Layout,”](#) on page 1428.
- 450. Added bits 25 and 26 to `feature_cap_mask` field in PCAM register. See [Table 953, “PCAM - Ports Capabilities Mask Register Fields,”](#) on page 1146

Draft C:

451. Updated OPC_MOD of UMR, SET_PSV and GET_PSV. See Table 44, “OPC_MOD Field Encoding,” on page 316.
452. Removed valid and pd from TLS Progress Context. See Table 262, “TLS Progress Params Context Layout,” on page 528.
453. Updated SET_PSV description to support TLS Progress Context. See Table 58, “SET_PSV Segment Format,” on page 327.
454. Updated resource_dump register. See Table 937, “RESOURCE_DUMP Register Layout,” on page 1130.
455. Removed *suspended* bit from state field in QP Context. See Table 20, “QP Context Field Descriptions,” on page 261.
456. Added ipsec_offload to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
457. Added IPSEC Capabilities. See Table 29.3.4.15, “IPSEC Capabilities,” on page 1789.
458. Added insert_trailer to Per Protocol Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
459. Added trailer_alignment, trailer_header_association and insert_trailer to ETH Segment. See Table 47, “Eth Segment Format,” on page 318.
460. Added ipsec_syndrome and ipsec_next_header to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
461. Added ipsec_syndrome and ipsec_next_header to Flow Table Entry Match Set Misc Parameters 2. See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013.
462. Added IPSEC_OFFLOAD bit to general_obj_types field in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
463. Added IPSEC Offload Object. See Section 8.27.27, “IPSEC Offload Object,” on page 606.
464. Added ACCESS_ASO bit to Opcode field in Ctrl Segment. See Table 43, “General Ctrl Segment Field,” on page 313.
465. Added ACCESS_ASO to OPC_MOD Table. See Table 44, “OPC_MOD Field Encoding,” on page 316.
466. Added ACCESS_ASO to WQE Construction Summary. See Table 106, “WQE Construction Summary,” on page 355.
467. Added ASO CTRL Segment. See Section 8.9.4.1.20, “ASO CTRL Segment,” on page 346.
468. Added ASO Data Segment. See Section 8.9.4.1.21, “ASO Data Segment,” on page 350.
469. Added ADD_ESP_TRANSPORT, L2_TO_L3_ESP_TUNNEL, ADD_ESP_TRANSPORT_OVER_UDP, DEL_ESP_TRANSPORT, L3_ESP_TUNNEL_TO_L2 and DEL_ESP_TRANSPORT_OVER_UDP bits to reformat_type field in Packet Reformat Context. See Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,” on page 2045.
470. Added reformat_add_esp_transport, reformat_l2_to_l3_esp_tunnel, reformat_add_esp_transport_over_udp, reformat_del_esp_transport, reformat_l3_esp_tunnel_to_l2 and reformat_del_esp_transport_over_udp to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
471. Removed dir_ing from CWTPM register. See Table 1240, “CWTPM - Congestion WRED ECN TClass and Pool Register Layout,” on page 1428.

Draft D:

472. Added `log_max_ipsec_offload` to IPSEC Capabilities. See Table 1611, “IPSEC Capabilities Layout,” on page 1790.
 473. Added `IPSEC_OFFLOAD` bit to action field in Flow Context. See Table 2149, “FLOW_CONTEXT Field Descriptions,” on page 2026.
 474. Added `ipsec_obj_id` to Flow Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.
 475. Added `ipsec_encrypt` and `ipsec_decrypt` to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
 476. Added IPSEC offload section. See Section 21.7, “IPsec Offload,” on page 1022.
 477. Added Resource Dump section. See Section 27.8, “Resource Dump,” on page 1062.
 478. Removed `vhca_tunnel_id` from `SEND_NVMF_CC`. See Table 1954, “SEND_NVMF_CC Input Structure Layout,” on page 1937.
 479. Removed `vhca_tunnel_id` from `QUERY_NVMF_CC`. See Table 1958, “QUERY_NVMF_CC_C_RESPONSE Input Structure Layout,” on page 1938.
 480. Removed `QUERY_NVMF_CC_RESPONSE` and `SEND_NVMF_CC` bits from `vhca_tunnel_commands` field in `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
 481. Added `boot_legacy_interrupt_disable` to `NV_ROM_BOOT_CONF1`. See Table 596, “NV_ROM_BOOT_CONF1 Layout,” on page 786.
 482. Added `boot_legacy_interrupt_disable_supported` to `NV_ROM_CAP`. See Table 590, “NV_ROM_CAP Layout,” on page 782.
 483. Added `uefi_proto_over_ctrl` to `NV_ROM_CAP`. See Table 590, “NV_ROM_CAP Layout,” on page 782.
 484. Added `uefi_hii_over_ctrl`, `uefi_fmp_over_ctrl` and `uefi_aip_over_ctrl` to `NV_ROM_INI`. See Table 588, “NV_ROM_INI Layout,” on page 778.
 485. Added `traffic_class` to TSAR Element. See Table 2588, “TSAR Element Attributes Format,” on page 2205.
 486. Removed MFSCR register from Management and Peripherals chapter.
 487. Added `string_tlv` field to Firmware Info. See Table 1403, “Firmware Info Layout,” on page 1548.
- Draft E:**
488. Added `umem_*_buffer_param_*` to VIRTIO Emulation Capabilities. Table 1615, “VIRTIO Emulation Capabilities Layout,” on page 1793.
 489. Added `umem_1`, `umem_2` and `umem_3` to `VIRTIO_Q`. See Table 306, “VIRTIO_Q Format,” on page 575.
 490. Removed `umem_id` and `umem_offset` bits from `modify_field_select` field in `VIRTIO_NET_Q`. See Table 305, “VIRTIO_NET_Q Fields Description,” on page 574.
 491. Added `nic_sq_scheduling`, `nic_bw_share`, `nic_rate_limit`, `log_nic_max_sched_depth` and `log_max_qos_nic_queue_group` to QoS Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
 492. Added NIC bit to `scheduling_hierarchy` field in `DESTROY_SCHEDULING_ELEMENT` Input command. See Table 2599, “DESTROY_SCHEDULING_ELEMENT Input Structure Field Descriptions,” on page 2207.

- 493. Added NIC bit to scheduling_hierarchy field in QUERY_SCHEDULING_ELEMENT Input command. See [Table 2603, “QUERY_SCHEDULING_ELEMENT Input Structure Field Descriptions,” on page 2209.](#)
- 494. Added NIC bit to scheduling_hierarchy field in CREATE_SCHEDULING_ELEMENT Input command. See [Table 2584, “CREATE_SCHEDULING_ELEMENT Input Structure Layout,” on page 2203.](#)
- 495. Added QUEUE_GROUP bit to element_type field in Scheduling Context. See [Table 2587, “Scheduling Context Field Descriptions,” on page 2204.](#)
- 496. Added qos_queue_group_id to SQ Context. See [Table 139, “SQ Context Fields,” on page 405](#)
- 497. Added qos_queue_group_id to Modify Bitmask Table Context. See [Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407.](#)
- 498. Changed the inline_data infinite array into a limited one in RESOURCE_DUMP. See [Table 937, “RESOURCE_DUMP Register Layout,” on page 1130.](#)

Draft F:

- 499. Added log_max_metadata_size to NVMe-oF Capabilities. See [Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.](#)
- 500. Added log_max_metadata_size to NVMe-oF Target Offload Context. See [Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.](#)
- 501. Added regexp_params to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 502. Added cc_roce_ecn_rp_classify_mode, cc_roce_ecn_rp_dynamic_rtt, cc_roce_ecn_rp_dynamic_ai, cc_roce_ecn_rp_dynamic_g and cc_roce_ecn_rp_burst_decouple to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 503. Added flow_classify_mode, dynamic_rtt_en, dynamic_rtt_qp_threshold_and_value_i, dynamic_ai_en, dynamic_ai, dynamic_g_en, dynamic_g_params and multi_qp_burst_decouple bits to field_select_R-RoCE_RP field in FIELD_SELECT_R-RoCE RP. See [Table 2679, “FIELD_SELECT_R-RoCE RP Structure Layout,” on page 2244.](#)
- 504. Added flow_classify_mode, dynamic_rtt_en, dynamic_ai_en, dynamic_g_en, multi_qp_burst_decouple, dynamic_g_inc_step, dynamic_g_dec_step, dynamic_rtt_qp_threshold_i, dynamic_rtt_value_i, max_dynamic_ai, min_dynamic_ai, dynamic_g_cnp_upper_threshold, dynamic_g_cnp_lower_threshold, max_dynamic_g and min_dynamic_g to Congestion Control R-RoCE ECN RP. See [Table 2671, “Congestion Control R-RoCE ECN RP Structure Layout,” on page 2235.](#)
- 505. Added roce_ecn_rp_dynamic_rtt_inc, roce_ecn_rp_dynamic_rtt_dec, roce_ecn_rp_dynamic_tcp_g_inc, roce_ecn_rp_dynamic_tcp_g_dec, roce_ecn_rp_dynamic_ai_inc, roce_ecn_rp_dynamic_ai_rst, roce_ecn_rp_cur_dce_tcp_rtt, roce_ecn_rp_cur_rpg_ai_rate, and roce_ecn_rp_cur_dce_tcp_g to QUERY_CONG_STATISTICS. See [Table 2687, “QUERY_CONG_STATISTICS Output Structure Layout,” on page 2247.](#)
- 506. Added NVME_DEVICE, VIRTIO_NET_DEVICE and VIRTIO_BLK_DEVICE bits to op_mod field in QUERY_EMULATED_FUNCTIONS_INFO command. See [Table 1963, “QUERY_EMULATED_FUNCTIONS_INFO Input Structure Field Descriptions,” on page 1940.](#)
- 507. Renamed GEN_OBJ_CMD_DEVICE_EUMULATION to GEN_OBJ_CMD_NVME_DEVICE_EUMULATION in vhca_tunnel_commands in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

- 508. Renamed `Device_Emulation_Capabilities` to `NVME_Device_Emulation_Capabilities` in `op_mode` in `QUERY_HCA_CAP`. See Table 1568, “`QUERY_HCA_CAP` Input Structure Field Descriptions,” on page 1667.
- 509. Renamed `VIRTIO_EMULATION_Capabilities` to `VDPA_EMULATION_Capabilities` in `op_mode` in `QUERY_HCA_CAP`. See Table 1568, “`QUERY_HCA_CAP` Input Structure Field Descriptions,” on page 1667.
- 510. Added `VIRTIO_NET_EMULATION_Capabilities`, `VIRTIO_BLK_EMULATION_Capabilities` and `HOTPLUG_Capabilities` bits to `op_mode` field in `QUERY_HCA_CAP`. See Table 1568, “`QUERY_HCA_CAP` Input Structure Field Descriptions,” on page 1667.
- 511. Added `VIRTIO_NET_EMULATION_Capabilities`, `VIRTIO_BLK_EMULATION_Capabilities` and `HOTPLUG_Capabilities` bits to `op_mode` field in `SET_HCA_CAP`. See Table 1628, “`SET_HCA_CAP` Input Structure Field Descriptions,” on page 1804.
- 512. Renamed `device_emulation_manager` to `nvme_device_emulation_manager` in `vhca_tunnel_commands` in `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 513. Added `desc_tunnel_offload_type` and `eth_frame_offload_type` to `VIRTIO_EMULATION` capabilities. See Table 1615, “`VIRTIO Emulation Capabilities Layout`,” on page 1793.
- 514. Added `hotplug_manager` to `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 515. Added `virtio_net_device_emulation_manager` and `virtio_blk_device_emulation_manager` to `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 516. Added Hotplug Capabilities. See Table 1619, “`Hotplug Capabilities Layout`,” on page 1798.
- 517. Added `DEVICE_OBJECT`, `VIRTIO_NET_DEVICE_EMULATION`, `VIRTIO_BLK_DEVICE_EMULATION` and `VIRTQ_BLK_Q` bits to `general_obj_types` field in `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 518. Added `VIRTIO NET DEVICE EMULATION` Object. See Section 8.27.19, “`VIRTIO NET DEVICE EMULATION` Object,” on page 582.
- 519. Added `VIRTIO BLK DEVICE EMULATION` Object. See Section 8.27.20, “`VIRTIO BLK DEVICE EMULATION` Object,” on page 587.
- 520. Renamed `cap_max_tclass_sb` to `cap_max_tclass_data` in `SBCAM` register. See Table 1324, “`SBCAM - Shared Buffer Capabilities Mask Register Layout`,” on page 1490.
- 521. Renamed `record` to `menu_record` in `Menu Segment`. See Section 840, “`Menu Segment Layout`,” on page 1073.
- 522. Renamed `index_1`, `index_2`, `num_of_obj_1` and `num_of_obj_2` to `index1`, `index2`, `num_of_obj1` and `num_of_obj2` respectively in `RESOURCE_DUMP` Register. See Section 938, “`RESOURCE_DUMP` Register Fields,” on page 1131.

Revision 0.49

Draft A:

- 523. Added `bsf_in_create_mkey` to `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 524. Added `roce_exp_backon`, `roce_tx_window` and `roce_slow_restart` to `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.
- 525. Added `uctx_cap` to `HCA_CAP`. See Table 1571, “`HCA Capabilities Layout`,” on page 1671.

526. Added `flow_counters_dump` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
527. Added bit `FLOW_METER` to *general_obj_types* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
528. Added `flex_parser_id_icmp_dw1`, `flex_parser_id_icmp_dw0`, `flex_parser_id_icmpv6_dw1`, `flex_parser_id_icmpv6_dw0`, `flex_parser_id_outer_first_mpls_over_gre` and `flex_parser_id_outer_first_mpls_over_udp_label` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
529. Added `ib_link_list_striding_wq` and `ib_cyclic_striding_wq` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
530. Added bits `flex_icmp_enabled` and `flex_icmpv6_enabled` to *flex_parser_protocols* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
531. Added `bsf_octword_actual_size` to `CREATE_MKEY` command. See [Table 1797, “CREATE_MKEY Input Structure Layout,” on page 1861](#).
532. Added `ROCE_ACCL` register. See [Section 28.3.7, “ROCE_ACCL - ROCE Accelerator,” on page 1091](#).
533. Increased the size of `num_of_counters` in `QUERY_FLOW_COUNTER` command from 15:0 to 29:0. See [Table 2182, “QUERY_FLOW_COUNTER Input Structure Layout,” on page 2041](#).
534. Added `dump_to_memory`, `mkey` and `address` to `QUERY_FLOW_COUNTER` command. See [Table 2182, “QUERY_FLOW_COUNTER Input Structure Layout,” on page 2041](#).
535. Added `esw_shared_ingress_acl` and `esw_uplink_ingress_acl` to E-switch capabilities. See [Table 1593, “E-switch Capabilities Layout,” on page 1768](#).
536. Added `table_vport_number` and `table_of_other_vport` to `SET_FLOW_TABLE_ROOT` command. See [Table 2112, “SET_FLOW_TABLE_ROOT Input Structure Layout,” on page 1995](#).
537. Added `outer_vxlan_gpe_vni`, `outer_vxlan_gpe_flags`, `outer_vxlan_gpe_next_protocol`, `inner_tcp_ack_num`, `outer_tcp_ack_num`, `inner_tcp_seq_num` and `outer_tcp_seq_num` to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761](#).
538. Added bits `OUT_TCP_ACK_NUM`, `IN_TCP_ACK_NUM`, `OUT_TCP_SEQ_NUM` and `IN_TCP_SEQ_NUM` to *field* field in `SET_ACTION`. See [Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051](#).
539. Added Flow Table Entry Match Set Misc Parameters 3 register. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,” on page 2016](#).
540. Added `inner_tcp_ack_num`, `outer_tcp_ack_num`, `inner_tcp_seq_num` and `outer_tcp_seq_num` to Flow Table Entry Match Set Misc Parameters 3. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,” on page 2016](#).
541. Added `outer_vxlan_gpe_vni`, `outer_vxlan_gpe_next_protocol` and `outer_vxlan_gpe_flags` to Flow Table Entry Misc Parameters 3. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,” on page 2016](#).
542. Added `icmp_header_data`, `icmpv6_header_data`, `icmp_type`, `icmp_code`, `icmpv6_type` and `icmpv6_code` to Flow Table Entry Misc Parameters 3. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,” on page 2016](#).

543. Added `icmp_header_data`, `icmpv6_header_data`, `icmp_type`, `icmp_code`, `icmpv6_type` and `icmpv6_code` to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,” on page 1761](#).
 544. Added `flex_parser_profile_2_supported` to `NV_FLEX_PARSER_CAP`. See [Table 400, “NV_FLEX_PARSER_CAP Layout,” on page 666](#).
 545. Added `pf_bar2_en` and `log_pf_bar2_size` to `NV_GLOBAL_PCI_CONF`. See [Table 430, “NV_GLOBAL_PCI_CONF Layout,” on page 682](#).
 546. Added `pf_bar2_supported` to `NV_GLOBAL_PCI_CAP`. See [Table 432, “NV_GLOBAL_PCI_CAP Layout,” on page 686](#).
 547. Added `log_esw_max_sched_depth` and `log_max_flow_meter` to Quality of Service Capabilities. See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773](#).
 548. Added `partial_reset` to `NV_POWER_CONF`. See [Table 562, “NV_POWER_CONF Layout,” on page 763](#).
 549. Added `partial_reset_supported` to `NV_POWER_CAP`. See [Table 564, “NV_POWER_CAP Layout,” on page 765](#).
 550. Added Flow Meter Object. See [Section 8.27.11, “Flow Meter Object,” on page 556](#).
 551. Added support for Shared Q Counters in `ALLOC_Q_COUNTER` command. See [Table 2553, “ALLOC_Q_COUNTER Input Structure Field Descriptions,” on page 2186](#).
 552. Added `esw_fdb_ipv4_ttl_modify` to Flow Table E-switch Capabilities Layout. See [Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748](#).
 553. Added `esw_fdb_ipv4_ttl_modify_enable` to `NV_SW_OFFLOAD_CONFIG`. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702](#).
 554. Added `esw_fdb_ipv4_ttl_modify_supported` to `NV_SW_OFFLOAD_CAP`. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707](#).
- Draft B:**
555. Added `gen3_cc_negotiation` to `HCA_CAP` Capabilities. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
 556. Added `gen3_cc_negotiation_en` to `QUERY_CONG_STATUS`. See [Table 2660, “QUERY_CONG_STATUS Output Structure Field Descriptions,” on page 2229](#).
 557. Added `gen3_cc_negotiation_en` and `gen3_cc_negotiation_mask` to `MODIFY_CONG_STATUS`. See [Table 2661, “MODIFY_CONG_STATUS Input Structure Layout,” on page 2230](#).
 558. Added Debug Capabilities. See [Table 1599, “Debug Capabilities Layout,” on page 1777](#).
 559. Added `log_cr_dump_to_mem_size` to Debug Capabilities. See [Table 1599, “Debug Capabilities Layout,” on page 1777](#).
 560. Added `CR_DUMP_TO_MEM` bit to `core_dump_type` field in `CORE_DUMP` Register. See [Table 935, “CORE_DUMP Register Layout,” on page 1128](#).
 561. Added `more_dump`, `mkey`, `address`, `size` and `cookie` to `CORE_DUMP` Register. See [Table 935, “CORE_DUMP Register Layout,” on page 1128](#).
 562. Added `log_max_extended_rnr_retry` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
 563. Added `extended_rnr_retry_valid`, `extended_log_rnr_retry` and `extended_cur_rnr_retry` to QP Context. See [Table 19, “QP Context Structure Layout,” on page 258](#).

- 564. Added `umem` bit to `resize_field_select` field in `Resize Fields Select`. See Table 1877, “`RESIZE_FIELD_SELECT` Structure Field Descriptions,” on page 1896.
- 565. Added `modify_field_select` and enabled to `Device Emulation Object`. See Table 270, “`NVMe Device Emulation Object Format`,” on page 541.
- 566. Added `cq_period_mode`, `cq_period` and `cq_max_count` to `NVMe CQ`. See Table 276, “`NVME_CQ Object Format`,” on page 547.
- 567. Resized `nvme_sq` in `SEND_NVMF_CC` to 32 bits. See Table 1954, “`SEND_NVMF_CC Input Structure Layout`,” on page 1937.
- 568. Resized `nvme_sq` in `QUERY_NVMF_CC` to 32 bits. See Table 1958, “`QUERY_NVMF_C-C_RESPONSE Input Structure Layout`,” on page 1938.
- 569. Removed `UCTX Object Context`.
- 570. Removed `UMEM Object Context`.
- 571. Renamed `Device Emulation Bar Registers Object` to `Device Emulation Object`. See Table 270, “`NVMe Device Emulation Object Format`,” on page 541.
- 572. Added `CREATE_UCTX`. See Section 29.23.5, “`CREATE_UCTX`,” on page 2087.
- 573. Added `QUERY_UCTX`. See Section 29.23.6, “`QUERY_UCTX`,” on page 2088.
- 574. Added `DESTROY_UCTX`. See Section 29.23.7, “`DESTROY_UCTX`,” on page 2090.
- 575. Added `CREATE_UMEM`. See Section 29.23.8, “`CREATE_UMEM`,” on page 2090.
- 576. Added `QUERY_UMEM`. See Section 29.23.9, “`QUERY_UMEM`,” on page 2093.
- 577. Added `DESTROY_UMEM`. See Section 29.23.10, “`DESTROY_UMEM`,” on page 2094.
- Draft C:**
- 578. Added `log_max_emulated_sq` and `log_max_emulated_cq` to `Device Emulation Capabilities`. See Table 1607, “`NVME Device Emulation Capabilities Layout`,” on page 1787.
- 579. Changed the opcode of `SEND_NVMF_CC` command. See Table 1550, “`Commands List Sorted by Opcode`,” on page 1647.
- 580. Changed the opcode of `QUERY_NVMF_CC_REPSPONSE` command. See Table 1550, “`Commands List Sorted by Opcode`,” on page 1647.
- 581. Added `Power Failure Event` subtype under `General Notification Event Subtype` table. See Table 215, “`General Notification Event Subtype`,” on page 496.
- 582. Added `ignore_flow_level` to `Flow Table Properties`. See Table 1587, “`Flow Table Properties Layout`,” on page 1753.
- 583. Added `ip_type` to `Flow Table Properties`. See Table 1587, “`Flow Table Properties Layout`,” on page 1753.
- 584. Added `MetaData Between Table Types` section. See Section 8.18.2.10.1, “`MetaData Between Table Types`,” on page 419.
- 585. Added `fdb_to_vport_reg_c_id` to `Flow Table E-switch Caps`. See Table 1585, “`Flow Table E-switch Capabilities Layout`,” on page 1748.
- 586. Added `fdb_to_vport_reg_c` and `fdb_to_vport_reg_c_id` to `Esw vport context`. See Table 27, “`Esw_vport Context Layout`,” on page 284.

587. Added `fdb_to_vport_reg_c_id` bit to *field_select* field in `MODIFY_ESW_VPORT_CONTEXT` command. See [Table 2500, “MODIFY_ESW_VPORT_CONTEXT Input Structure Field Descriptions,”](#) on page 2160.
588. Added `flex_geneve_tlv_option_0_enabled` bit to *flex_parser_protocols* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
589. Added `flex_parser_id_geneve_tlv_option_0` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
590. Added `max_geneve_tlv_option_data_len` and `max_geneve_tlv_options` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
591. Added `GENEVE_TLV_OPTION` bit to *general_obj_types* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
592. Added `ENCRYPTION_KEY` bit to *general_obj_types* field in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
593. Added `GENEVE_TLV_OPTION` Object. See [Section 8.27.13, “GENEVE_TLV_OPTION Object,”](#) on page 562.
594. Added `DEK` Object. See [Section 8.27.14, “DEK Object,”](#) on page 564.
595. Added `geneve_tlv_option_0_data` to `Flow Table Fields Supported`. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
596. Added `geneve_tlv_option_0_data` to `Flow Table Entry Match Set Misc parameters 3`. See [Table 2134, “Flow Table Entry Match Set Misc Parameters 3 Format,”](#) on page 2016.
597. Added `force_entropy_cap`, `entropy_calc_cap`, `entropy_gre_calc_cap`, `force_entropy_calc` and `entropy_gre_calc` to `PCMR` register. See [Table 1167, “PCMR - Ports Check Mask Register Layout,”](#) on page 1357.
598. Added `max_flow_meter_bs_exponent` to `QoS Capabilities`. See [Table 1597, “Quality of Service Capabilities Layout,”](#) on page 1773.
599. Added `high_temperature` bit to *syndrome* field in `FPGA error event`. See [Table 241, “Event - data Field - FPGA Error Event Field Descriptions,”](#) on page 504.
600. Added `error_type` to `FPGA_CTRL`. See [Table 895, “FPGA_CTRL Register Layout,”](#) on page 1108.
601. Added `inner_ip_type` and `outer_ip_type` fields to `QUERY_FLOW_GROUP` command. See [Table 2144, “QUERY_FLOW_GROUP Output Structure Layout,”](#) on page 2021.
602. Added `inner_ip_type` and `outer_ip_type` fields to `CREATE_FLOW_GROUP` command. See [Table 2120, “CREATE_FLOW_GROUP Input Structure Layout,”](#) on page 2000.
603. Added `TLS Offload` section. See [Section 4.9, “TLS Offload,”](#) on page 216.
604. Added `tls` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
605. Added `TLS Capabilities`. See [Section 29.3.4.14, “TLS Capabilities,”](#) on page 1788.
606. Added `TLS Static Params Context`. See [Section 8.22, “Transport Static Params Context,”](#) on page 525.
607. Added `TLS Progress Params Context`. See [Section 8.23, “TLS PROGRESS Params Context,”](#) on page 528.
608. Added `tls_en` to `TIR Context`. See [Table 116, “TIR Context Format,”](#) on page 374.
609. Added `tls_en` to `TIS Context`. See [Table 121, “TIS Context Format,”](#) on page 382.

- 610. Added `log_max_dek` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 611. Added `UMR`, `GET_PSV` and `SET_PSV` entries under `OPC_MOD` table. See Table 44, “`OPC_MOD` Field Encoding,” on page 316.
- 612. Renamed `immediate_or_invalidation_key_or_umr_mkey_or_remote_micro_app_operation_code` to `ctrl_general_id` in `Ctrl Segment`. See Table 43, “General Ctrl Segment Field,” on page 313.
- 613. Added `outer_gre_c_present`, `outer_gre_k_present` and `outer_gre_s_present` to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- 614. Added `gre_c_present`, `gre_k_present` and `gre_s_present` to Flow Table Entry Match Set Misc Parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- 615. Added `wqe_index_ignore_cap` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 616. Renamed `uctx_cap` to `uctx_permission` in `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 617. Re-added the following mistakenly deleted sections: Section 14.7, “vNIC Monitoring,” on page 856, Section 14.8, “Multi vHCA Mkey,” on page 856, and Section 14.9, “Multi PCI Port,” on page 857.
- Draft D:**
- 618. Added Encapsulation Entropy section. See Section 8.18.3.7, “Encapsulation Entropy,” on page 423.
- 619. Added `flow_meter_reg_id` to QoS Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
- 620. Added `sw_steering_vport_icm_address_rx` and `sw_steering_vport_icm_address_tx` to Esw vport Context. See Table 27, “Esw_vport Context Layout,” on page 284.
- 621. Added `sw_owner_icm_root_0` and `sw_owner_icm_root_1` to Flow Table Context. See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989.
- 622. Added `sw_steering_fdb_action_drop_icm_address_rx`, `sw_steering_fdb_action_drop_icm_address_tx`, `sw_steering_uplink_icm_address_rx` and `sw_steering_uplink_icm_address_tx` to Flow Table Eswitch Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748.
- 623. Added `flow_counter_bulk_alloc` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 624. Added `flow_counter_bulk` to `ALLOC_FLOW_COUNTER`. See Table 2174, “`ALLOC_FLOW_COUNTER` Input Structure Layout,” on page 2038.
- 625. Added `vf_nodnic_enable` to `NV_GLOBAL_PCI_CONF`. See Table 430, “`NV_GLOBAL_PCI_CONF` Layout,” on page 682.
- 626. Added `vf_nodnic_supported` to `NV_GLOBAL_PCI_CAP`. See Table 432, “`NV_GLOBAL_PCI_CAP` Layout,” on page 686.
- 627. Added DPP section. See Section , “The keys must be destroyed after the steering entry has been changed. In case of send flow, Keys can only be removed after completion of send WQE.,” on page 220.
- 628. Added `dpp` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- 629. Added DPP Capabilities. See Table 1613, “DPP Capabilities Layout,” on page 1792.
- 630. Added DPP bit to mem_rq_type field in RQ Context Format. See Table 124, “RQ Context Format,” on page 385.
- 631. Added dpp_wire_protocol, dpp_mkey, dpp_buffer_size and dpp_segment_size to RQ Context Format. See Table 124, “RQ Context Format,” on page 385.
- 632. Removed host_params from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 633. Added sf_base_id to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 634. Added esw_functions_changed, log_max_esw_sf and esw_sf_base_id to E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
- 635. Renamed QUERY_HOST_PARAMS to QUERY_ESW_FUNCTIONS. See Section 29.29, “QUERY_ESW_FUNCTIONS - Query e-Switch Functions,” on page 2140.
- 636. Renamed HOST_PARAMS_CHANGE to ESW_FUNCTIONS_CHANGED. See Section 8.20.14.4, “ESW_FUNCTIONS_CHANGED Event,” on page 503.
- 637. Removed flow_counter_id and modify_header_id from Flow Meter Object. See Table 286, “Flow Meter Object Format,” on page 556.
- 638. Added table_type, destination_table_id, sw_steering_icm_address_rx and sw_steering_icm_address_tx to Flow Meter Object. See Table 286, “Flow Meter Object Format,” on page 556.
- 639. Renamed flow_meter_srctm to flow_meter in QoS Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
- 640. Removed byte_token_ratio from Flow Meter Parameters. See Table 288, “Flow Meter Parameters Format,” on page 558.
- 641. Added meter_mode to Flow Meter Parameters. See Table 288, “Flow Meter Parameters Format,” on page 558.
- 642. Added network_state to NVME_SQ. See Table 272, “NVME_SQ Object Format,” on page 543.
- 643. Added QUERY_EMULATED_FUNCTIONS_INFO. See Section 29.10.19, “QUERY_EMULATED_FUNCTIONS_INFO - Query Emulated Functions Information,” on page 1940.
- 644. Added total_emulated_pfs to Device Emulation Caps. See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787.
- 645. Removed NVMe_emulation_to_NVMf bit from offload_type field in QP Context. See Table 20, “QP Context Field Descriptions,” on page 261.
- 646. Removed cmd_on_behalf and function_id from INIT_HCA. See Table 1666, “INIT_HCA Input Structure Layout,” on page 1820.
- 647. Removed cmd_on_behalf from ENABLE_HCA. See Table 1674, “ENABLE_HCA Input Structure Layout,” on page 1823.
- 648. Removed cmd_on_behalf from DISABLE_HCA. See Table 1678, “DISABLE_HCA Input Structure Layout,” on page 1825.
- 649. Added NVMe Emulation Offload section. See Section 20.4.4, “NVMe Emulation Offload,” on page 995.
- 650. Added NV_EMULATION_NVME_CONF. See Section 11.7.97, “NV_EMULATION_NVME_CONF,” on page 767.

- 651. Added NV_EMULATION_NVME_CAP. See [Section 11.7.98, “NV_EMULATION_NVME_CAP,”](#) on page 768.
- 652. Added new register MERR - Management Entities Remapping Register. See [Section 28.26.26, “MERR - Management Entities Remapping Register,”](#) on page 1569.
- 653. Added sdm and host_buses to MPIR - Management PCIe Index Register. See [Table 1500, “MPIR - Management PCIe Index Register Layout,”](#) on page 1606.
- 654. Added pwr_status and pci_power to MPEIN - Management PCIE INfo Register. See [Table 1488, “MPEIN - Management PCIE INfo Register Layout,”](#) on page 1593.
- 655. Updated description of pcie_index in MPEIN - Management PCIE INfo Register. See [Table 1488, “MPEIN - Management PCIE INfo Register Layout,”](#) on page 1593.
- 656. Added a new description to power_consumption field in MSPS Register. See [Table 1357, “MSPS - Management System Power Supply Register Layout,”](#) on page 1516.
- 657. Added new register PHCR - Port Histogram Configuration Register. See [Table 1151, “PHCR - Port Histogram Configuration Register Layout,”](#) on page 1337.
- 658. Added a new bit to PPAOS for Force port down.
- Draft E:**
- 659. Added table_miss_action_domain to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- 660. Added termination_table to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- 661. Added switch_domain bit to table_miss_action field in Flow Table Context. See [Table 2103, “FLOW TABLE CONTEXT Field Descriptions,”](#) on page 1990.
- 662. Added PCI_POWER_CHANGE_EVENT. See [Table 220, “Event_data Field - General Notification PCI Power Change Event Layout,”](#) on page 497.
- 663. Added geneve_tlv_option_0_exist to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- 664. Added geneve_tlv_option_0_exist to Flow Table Entry Match Set Misc parameters. See [Table 2130, “Flow Table Entry Match Set Misc Parameters Format,”](#) on page 2009.
- 665. Added strict_vf_msix_num_enabled to NV_GLOBAL_PCI_CONF. See [Table 430, “NV_GLOBAL_PCI_CONF Layout,”](#) on page 682.
- 666. Added strict_vf_msix_num_supported to NV_GLOBAL_PCI_CAP. See [Table 432, “NV_GLOBAL_PCI_CAP Layout,”](#) on page 686.
- 667. Added sig_block_4048 to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 668. Added Index 6 to Block Size Table. See [Table 702, “Block Size Table,”](#) on page 912.
- 669. Removed esw_fdb_flow_table_root_per_port from Flow table Eswitch Caps. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
- 670. Added root_ft_on_other_esw to E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- 671. Added table_eswitch_owner_vhca_id_valid and table_eswitch_owner_vhca_id to SET_FLOW_TABLE_ROOT command. See [Table 2112, “SET_FLOW_TABLE_ROOT Input Structure Layout,”](#) on page 1995.

- 672.Updated ROCE_ACCL register. See [Table 868, “ROCE_ACCL Register Layout,” on page 1092.](#)
- 673.Added termination_table to Flow Table Context. See [Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989.](#)
- 674.Added ignore_flow_level to SET_FLOW_TABLE_ENTRY. See [Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023.](#)
- 675.Added VIRTQ_NET bit to general_obj_types field in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 676.Added VIRTIO Emulation Capabilities. See [Section 29.3.4.17, “VIRTIO Emulation Capabilities,” on page 1792.](#)
- 677.Added VIRTIO_NET_Q Object. See [Section 8.27.18, “VIRTIO_NET_Q Object,” on page 571.](#)
- 678.Added inline_q_type to TIR context. See [Table 116, “TIR Context Format,” on page 374.](#)
- 679.Added list_q_type to RQT context. See [Table 131, “RQT Context Fields,” on page 400.](#)
- 680.Added prio_tag_required_en to NV_SW_OFFLOAD_CONFIG. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.](#)
- 681.Added prio_tag_required_supported to NV_SW_OFFLOAD_CAP. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707](#)
- 682.Added prio_tag_required to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 683.Added PHCR - Ports Histogram Configuration Register. See [Section 28.19.54, “PHCR - Ports Histogram Configuration Register,” on page 1337.](#)

Draft F:

- 684.Removed roce_adp_retrans, roce_tx_window and roce_slow_restart from HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 685.Added roce_accl to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 686.Added uctx_en to NV_SW_OFFLOAD_CONFIG. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.](#)
- 687.Added uctx_supported to NV_SW_OFFLOAD_CAP. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707.](#)
- 688.Added reformat_and_fwd_to_table to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,” on page 1753.](#)
- 689.Added dynamic_vf_msix to NV_PCI_CONF. See [Table 404, “NV_PCI_CONF Layout,” on page 668.](#)
- 690.Added dynamic_vf_msix_supported to NV_PCI_CAP. See [Table 406, “NV_PCI_CAP Layout,” on page 670.](#)

Draft G:

- 691.Added dynamic_msix_table_size, num_total_dynamic_vf_msix, max_dynamic_vf_msix_table_size and min_dynamic_vf_msix_table_size to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 692.Added lag_tx_port_affinity to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- 693.Added host_total_vfs to Host Params Context. See [Table 2451, “HOST_PARAMS Context Layout,” on page 2142.](#)

- 694.Updated the Resiliency and Error Recovery section. See [Section 12.12, “Resiliency and Error Recovery,”](#) on page 829.
- 695.Added `cqe_checksum_full` to Per Protocol Networking Caps. See [Section 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- 696.Added `forward_vhca_rx` and `forward_vhca_tx` to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
- 697.Added `VHCA_RX` and `VHCA_TX` bits to `destination_type` field in Destination Entry. See Table 2159, “Destination Entry Format Field Descriptions,” on page 2032.
- 698.Added `VHCA_TUNNEL_OBJECT` bit to `general_obj_types` field in `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 699.Added `vHCA_Tunnel` Object. See [Section 8.27.26, “VHCA Tunnel Object,”](#) on page 605
- 700.Added `vhca_tunnel_id` to `INIT_HCA`, `TEARDOWN_HCA`, `ENABLE_HCA`, `DISABLE_HCA`, `QUERY_SPECIAL_CONTEXT`, Flow Table Commands, QP Commands, `SEND_NVMF_CC`, `QUERY_NVMF_CC_RESPONSE`, `GENERAL_OBJ` commands, `SET_ROCE_ADDRESS`, `QUERY_ROCE_ADDRESS` commands.
- 701.Removed `cmd_on_behalf` from `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 702.Added `vhca_tunnel_commands` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- 703.Resized `nvme_cq_id` in `NVME_SQ` to 32 bits. See Table 272, “NVME_SQ Object Format,” on page 543.
- 704.Removed `mkey_buffer_umem_id` and `mkey_buffer_umem_offset` from `NVME_SQ` to 32 bits. See Table 272, “NVME_SQ Object Format,” on page 543.
- 705.Added `qpn_vhca_id` to `NVME_SQ`. See Table 272, “NVME_SQ Object Format,” on page 543.
- 706.Replaced `vhca_id` with `device_emulation_id` in `NVME_SQ`. See Table 272, “NVME_SQ Object Format,” on page 543.
- 707.Replaced `vhca_id` with `device_emulation_id` in `NVME_CQ`. See Table 276, “NVME_CQ Object Format,” on page 547.
- 708.Removed `nvme_namespace` and `num_of_namespaces` from `NVME_SQ`. See Table 272, “NVME_SQ Object Format,” on page 543.
- 709.Added NVME NAMESPACE OBJECT. See [Section 8.27.5, “NVME_NAMESPACE OBJEC,”](#) on page 546.
- 710.Updated NVMe Emulation Offload. See [Section 20.4.4, “NVMe Emulation Offload,”](#) on page 995.
- 711.Added `NV_GLOBAL_PCI_CONF_2` TLV. See [Section 11.7.31, “NV_GLOBAL_PCI_CONF_2,”](#) on page 694.
- 712.Added `pcie_credit_token_timeout_supported` to `NV_GLOBAL_PCI_CAP`. See [Section 11.7.29, “NV_GLOBAL_PCI_CAP,”](#) on page 686.
- 713.Added PCIe Hanged Event section. See [Section 27.7, “PCIe Hanged Event,”](#) on page 1061.
- 714.Added `pcie_hanged` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- 715. Added `Pcie_Hanged_Event` to General Notification event. See [Table 215, “General Notification Event Subtype,”](#) on page 496.
- 716. Added `roce_adp_retrans`, `roce_adp_retrans_to`, `roce_slow_restart`, `roce_slow_restart_cnps` and `roce_slow_restart_trans` to `QUERY_Q_COUNTER`. See [Table 2562, “QUERY_Q_COUNTER Output Structure Layout,”](#) on page 2188.
- 717. Added `roce_adp_retrans_rtt` to QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- 718. Added `pd` to TIS context. See [Table 121, “TIS Context Format,”](#) on page 382
- 719. Added `tls_offload` to CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
- 720. Added `pd` to DEK Object. See [Table 294, “DEK Object Format,”](#) on page 564.
- 721. Added `log_rmp_size` to NVMe-oF Target Offload Context. See [Table 256, “NVMe-oF Target Offload Context Header Layout,”](#) on page 519.
- 722. Added `mode` to PHBR register. See [Table 1146, “PHBR - Port Histogram Binding Register Layout,”](#) on page 1334.
- 723. Added new Port Module Type Mapping Register (PMTM) register. See [Table 1229, “PMTM - Port Module Type Mapping Register Layout,”](#) on page 1418.
- 724. Added new Port Local Allocation Register (PLAR) register. See [Table 1231, “PLAR- Port Local Allocation Register Layout,”](#) on page 1419.
- Draft H:**
- 725. **Removed `QUERY_VIRTIO_EMULATION_INFO` command.**
- 726. Added `host_pf_disabled`, `host_pf_vhca_id_valid` and `host_pf_vhca_id` to Host Params Context. See [Table 2451, “HOST_PARAMS Context Layout,”](#) on page 2142.
- 727. Added `lag_native_fdb_selection` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 728. Added `fdb_selection_mode` to LAG Context. See [Table 1762, “LAG CONTEXT Structure Layout,”](#) on page 1851.
- 729. Added `counter_set_id` to `NVME_SQ`. See [Table 272, “NVME_SQ Object Format,”](#) on page 543.
- 730. Added `counter_set_id` to Device Emulation Object. See [Table 270, “NVMe Device Emulation Object Format,”](#) on page 541.
- 731. Added `counter_set_id` to `NVME_NAMESPACE` Object. See [Table 274, “NVME_NAMESPACE Object Format,”](#) on page 546.
- 732. Added `NVME_CTRL_COUNTERS` Object. See [Section 8.27.7, “NVME_CTRL_COUNTERS Object,”](#) on page 549.
- 733. Added `NVME_NAMESPACE_COUNTERS` Object. See [Section 8.27.8, “NVME_NAMESPACE_COUNTERS Object,”](#) on page 551.
- 734. Added `NVME_SQ_COUNTERS` Object. See [Section 8.27.9, “NVME_SQ_COUNTERS Object,”](#) on page 552.
- 735. Added Object Change Event. See [Section 8.20.9, “Object Change Event,”](#) on page 481.
- 736. Added `event_cap` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- 737. Added Event Capabilities. See [Section 29.3.4.18, “Event Capabilities,”](#) on page 1795.

- 738. Resized `event_bitmask` in `CREATE_EQ` command. See Table 1843, “`CREATE_EQ` Input Structure Layout,” on page 1880.
- 739. Added `virtual_node_guid_en` and `virtual_node_guid` to `ICMD_QUERY_VIRTUAL_MAC`. See Table 2754, “`ICMD_QUERY_VIRTUAL_MAC` Output Mailbox Layout,” on page 2285.
- 740. Added `virtual_node_guid_en` and `virtual_node_guid` to `ICMD_SET_VIRTUAL_MAC`. See Table 2756, “`ICMD_SET_VIRTUAL_MAC` Input Mailbox Layout,” on page 2287.
- 741. Added `virtual_node_guid` to `ICMD_QUERY_CAPABILITY`. See Table 2736, “`ICMD_QUERY_CAPABILITY` General Capability Output Mailbox Layout,” on page 2274.
- 742. Added `vendor_id` and `subsystem_vendor_id` to Device Emulation Object. See Table 270, “`NVMe` Device Emulation Object Format,” on page 541.
- 743. Added `system_image_guid_modifiable` to `HCA_CAP`. See Table 1571, “`HCA` Capabilities Layout,” on page 1671.
- 744. Updated `MODIFY_NIC_VPORT_CONTEXT` Bitmask to support modifying `system_image_guid`. See Table 35, “`MODIFY_NIC_VPORT_CONTEXT` Bitmask,” on page 293.
- 745. Added `system_image_guid` bit to `field_select` field in `HCA_Vport` Context. See Table 37, “`HCA_Vport` Context Field Descriptions,” on page 296.

Revision 0.48 (October 2018)

Draft A:

1. Added CE to CQE Format, See Table 145, “`64B` Completion Queue Entry Format Layout,” on page 435
2. Added `metadata_size` to `ATTACH_NVMF_NAMESPACE`. See Table 2433, “`ATTACH_NVMF_NAMESPACE` Input Structure Layout,” on page 2135.
3. Added `max_lba_metadata_size` to NVMe capabilities. See Table 1603, “`NVMe-oF` Capabilities Layout,” on page 1780.
4. Added `nic_rx_flow_tag_multipath_en` to Flow Table Nic Capabilities. See Table 1583, “`Flow Table NIC` Capabilities Layout,” on page 1742.
5. Added `insert_always` to `vport_cvlan_insert` in `ESW_VPORT_CONTEXT`. See Table 28, “`Esw_vport` Context Field Descriptions,” on page 286.
6. Added `vport_cvlan_insert_always` to `ESWITCH` Capabilities. See Table 1593, “`E-switch` Capabilities Layout,” on page 1768.
7. Added `per_type_modifier` field for “`STRS_FAULT_INJECT` Register” on page 1122.
8. Added `SMBus` Failed Fault Inject as `per_type_modifier`. See Table 923, “`SMBUS_FAILED` Fault Inject Modifier Layout,” on page 1124.
9. Added `SXP` Hang Stop Toggle Modifier. See Table 907, “`SXP_HANG` Stop Toggle Modifier Layout,” on page 1117.
10. Added `port_number` enumeration in `RXH` Hang Stop Toggle Modifier. See Table 905, “`RXB_HANG` Stop Toggle Modifier Layout,” on page 1116.
11. Added `IRISC` Hang Mini-Flow Modifier. See Table 927, “`IRISC_HANG` Mini-Flow Modifier Layout,” on page 1125.

12. Added Packet Drop Mini-Flow Modifier. See Table 929, “PACKET_DROP Mini-Flow Modifier Layout,” on page 1126.
13. Added IRISC_HANG and PACKET_DROP mini-flow type in STRS Mini-Flow Register. See Table 925, “STRS_MINI_FLOW Register Layout,” on page 1124.
14. Added per_type_modifier field for STRS Mini-Flow Register. See Table 925, “STRS_MINI_FLOW Register Layout,” on page 1124.
15. Added Dump and Fill Memory Key. See [Section 18.2, “Dump and Fill Memory Key,”](#) on page 917.
16. Added dump_fill_mkey to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
17. Added sw_owner to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
18. Added steering_sw_icm_start_address to Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
19. Added log_steering_sw_icm_size to Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
20. Added log_sw_icm_alloc_granularity to Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
21. Added sw_steering_nic_rx_action_drop_icm_address to Flow Table NIC Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
22. Added SW_ICM to access_mode in Mkey Context. See Table 363, “MKey Context Fields,” on page 631.
23. Added sw_owner and sw_owner_icm_root to Flow Table Context. See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989.
24. Added icm_address to CREATE_TIR Output. See Table 1990, “CREATE_TIR Output Structure Layout,” on page 1949.
25. Added icm_address to CREATE_FLOW_TABLE Output. See Table 2100, “CREATE_FLOW_TABLE Output Structure Layout,” on page 1988.
26. Added SW ICM Block Object. See [Section 8.27.10, “SW ICM Block Object,”](#) on page 554.
27. Removed pci_atomic_operation and pci_atomic_size from Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728.
28. Added fetch_add_pci_atomic, swap_pci_atomic and compare_swap_pci_atomic to Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728.
29. Added boot_option_rom_en to NV_ROM_BOOT_CONF1. See [Table 596, “NV_ROM_BOOT_CONF1 Layout,”](#) on page 786.

Draft B:

30. Changed Bitmask of max_qp_retry_limit in Modify NIC Vport Context Bitmask. See [Table 35, “MODIFY_NIC_VPORT_CONTEXT Bitmask,”](#) on page 293.
31. Added suspend_qp_rc, suspend_qp_ud and suspend_qp_uc to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
32. Added reformat_and_modify_action to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.

33. Renamed `encap` to `reformat` in Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,” on page 1753.](#)
34. Renamed `encap_and_vlan` to `reformat_and_vlan` in Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,” on page 1753.](#)
35. Renamed `log_max_encap_headers` to `log_max_packet_reformat_context` in Flow Table NIC Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.](#)
36. Renamed `log_max_encap_headers` to `log_max_packet_reformat_context` in Flow Table E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,” on page 1768.](#)
37. Renamed `encap_en` to `reformat_en` in Flow Table Context. See [Table 2102, “FLOW_TABLE_CONTEXT Structure Layout,” on page 1989.](#)
38. Renamed `encap_id` to `packet_reformat_id` in Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.](#)
39. Renamed `ENCAP` bit to `REFORMAT` in Action field in Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.](#)
40. Renamed `encap_id` bit to `packet_reformat_id` in `modify_enable_mask` field in Set Flow Table Entry Command. See [Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023.](#)
41. Added `reformat_l3_tunnel_to_l2` and `reformat_l2_to_l3_tunnel` to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,” on page 1753.](#)
42. Renamed `ALLOC_ENCAP_HEADER` to `ALLOC_PACKET_REFORMAT_CONTEXT`. See [Section 29.18.15, “ALLOC_PACKET_REFORMAT_CONTEXT - Allocate Packet Reformat Context,” on page 2043.](#)
43. Renamed `DEALLOC_ENCAP_HEADER` to `DEALLOC_PACKET_REFORMAT_CONTEXT`. See [Section 29.18.16, “DEALLOC_PACKET_REFORMAT_CONTEXT - De-Allocate Packet Reformat Context,” on page 2046.](#)
44. Renamed `QUERY_ENCAP_HEADER` to `QUERY_PACKET_REFORMAT_CONTEXT`. See [Section 29.18.17, “QUERY_PACKET_REFORMAT_CONTEXT – Query Packet Reformat Context,” on page 2047.](#)
45. Renamed `encap_header` to `packet_reformat_context` in `ALLOC_PACKET_REFORMAT_CONTEXT`. See [Table 2186, “ALLOC_PACKET_REFORMAT_CONTEXT Input Structure Layout,” on page 2043.](#)
46. Renamed `encap_id` to `packet_reformat_id` in `ALLOC_PACKET_REFORMAT_CONTEXT`. See [Table 2188, “ALLOC_PACKET_REFORMAT_CONTEXT Output Structure Layout,” on page 2044.](#)
47. Renamed `Header_type` to `reformat_type` in `PACKET_REFORMAT_CONTEXT`. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,” on page 2045.](#)
48. Renamed `VXLAN` bit to `L2_TO_VXLAN` in `reformat_type` field in `PACKET_REFORMAT_CONTEXT`. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,” on page 2045.](#)
49. Renamed `NVGRE` bit to `L2_TO_NVGRE` in `reformat_type` field `PACKET_REFORMAT_CONTEXT`, See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,” on page 2045,](#)

50. Renamed GENERAL_HEADER bit to L2_TO_L2_TUNNEL in reformat_type field in PACKET_REFORMAT_CONTEXT. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,”](#) on page 2045.
51. Renamed encap_header_size to reformat_data_size in PACKET_REFORMAT_CONTEXT. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,”](#) on page 2045.
52. Renamed encap_header to reformat_data in PACKET_REFORMAT_CONTEXT. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,”](#) on page 2045.
53. Added L3_TUNNEL_TO_L2 and L2_TO_L3_TUNNEL bits to reformat_type field in PACKET_REFORMAT_CONTEXT. See [Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,”](#) on page 2045.
54. Added ICMD_QUERY_CAPABILITY Output Mailbox. See [Table 2734, “ICMD_QUERY_CAPABILITY Output Mailbox Layout,”](#) on page 2273.
55. Added ICMD_SET_DIAGNOSTIC_PARAMS. See [Section 30.6.1, “ICMD_SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters,”](#) on page 2292.
56. Added ICMD_QUERY_DIAGNOSTIC_PARAMS. See [Section 30.6.2, “ICMD_QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters,”](#) on page 2293.
57. Added ICMD_QUERY_DIAGNOSTIC_COUNTERS. See [Section 30.6.3, “ICMD_QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters,”](#) on page 2294.
58. Added capability_groups to ICMD_QUERY_CAPABILITY General Capability Output Mailbox. See [Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,”](#) on page 2274.
59. Added ICMD_QUERY_CAPABILITY Debug Capability Output Mailbox. See [Table 2738, “ICMD_QUERY_CAPABILITY Debug Capability Output Mailbox Layout,”](#) on page 2276.
60. Added Debug Capability bit to capability_group field in ICMD_QUERY_CAPABILITY Input Mailbox. See [Table 2733, “ICMD_QUERY_CAPABILITY Input Mailbox Field Description,”](#) on page 2273.
61. Added tracer_dump to DIAGNOSTIC_PARAMS_CONTEXT Input Structure. See [Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,”](#) on page 2223.
62. Added diag_counter_tracer_dump to Debug Capabilities. See [Table 1599, “Debug Capabilities Layout,”](#) on page 1777.
63. Added SYNC_STEERING. See [Section 29.18.21, “SYNC_STEERING – Synchronize Steering Tables,”](#) on page 2057.

Draft C:

64. Added max_num_eqs to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
65. Updated the description of log_max_eq in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
66. Added extended_fdb_flow_counter_en to Flow Table E-switch Capabilities. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
67. Added dbr_umem_valid to QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
68. Added dbr_umem_valid to WQ Context, See [Table 126, “Work Queue \(WQ\) Format,”](#) on page 389

69. Added `wq_umem_valid` to WQ Context. See Table 126, “Work Queue (WQ) Format,” on page 389.
70. Added `dbr_umem_valid` to CQ Context. See Table 171, “Completion Queue Context Layout,” on page 467.
71. Added `mkey_umem_valid` to CREATE_MKEY. See Table 1797, “CREATE_MKEY Input Structure Layout,” on page 1861.
72. Added `cq_umem_valid` to CREATE_CQ. See Table 1860, “CREATE_CQ Input Structure Layout,” on page 1888.
73. Added `wq_umem_valid` to CREATE_QP. See Table 1881, “CREATE_QP Input Structure Layout,” on page 1899.
74. Updated Section 11, “Non-Volatile Configuration (NVConfig),” on page 642.
75. Added NV Configurations Privileges section. See Section 11.3, “NV Configurations Privileges,” on page 643.
76. Added NV_HOST_PRIV_CONF TLV. See Section 11.7.18, “NV_HOST_PRIV_CONF,” on page 673.
77. Added NV_HOST_PRIV_CAP TLV. See Section 11.7.19, “NV_HOST_PRIV_CAP,” on page 674.

Draft D:

78. Added `qp_packet_based` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
79. Added `req_e2e_credit_mode` to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
80. Added `sw_steering_nic_tx_action_drop_icm_address` and `sw_steering_nic_tx_action_allow_icm_address` to Flow Table NIC Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
81. Added new supported objects to `general_obj_types` in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
82. Added `ecpf_vport_exists` to E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
83. Resized `log_steering_sw_icm_size` in Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
84. Added `packet_reformat` to Destination Entry. See Table 2158, “Destination Entry Format Structure Layout,” on page 2031.
85. Added Extended Destination Entry. See Table 2162, “Extended Destination Format Structure Layout,” on page 2033.
86. Added Extended Flow Counter List Entry. See Table 2164, “Extended Flow Counter List Entry Format Structure Layout,” on page 2033.
87. Updated Section 8.18.3.6, “Allocating a Packet Reformat Context,” on page 422.
88. Added `extended_destination` to Flow Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.
89. Added `log_max_fdb_encap_uplink` to E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.

90. Added Destination and Counter Lists. See [Table 2154, “Destination and Counter Lists Structure Layout,” on page 2030.](#)

91. Added Extended Destination and Counter Lists. See [Table 2156, “Extended Destination and Counter Lists Structure Layout,” on page 2030.](#)

Draft E:

92. Added IB Port Sniffer commands. See [Section 29.38, “IB Port sniffer,” on page 2217.](#)

93. Added `ib_port_sniffer` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

94. Added `xrc_srq_umem_offset`, `xrc_srq_umem_id` and `xrc_srq_umem_valid` to `CREATE_XRC_SRQ` Command. See [Table 2366, “CREATE_XRC_SRQ Input Structure Layout,” on page 2110.](#)

95. Added `cq_umem_offset`, `cq_umem_id` and `cq_umem_valid` to `MODIFY_CQ` Command. See [Table 1872, “MODIFY_CQ Input Structure Layout,” on page 1894.](#)

96. Added `function_id` and `cmd_on_behalf` to `CREATE_MKEY` command. See [Table 1797, “CREATE_MKEY Input Structure Layout,” on page 1861.](#)

97. Added `lag_dct` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

98. Added `CREATE_DCT_LAG` command. See [Section 29.30.6, “CREATE_DCT_LAG – Create DCT LAG,” on page 2150.](#)

Draft F:

99. Renamed ***fast_tearardown*** to ***hard_coded_1*** in `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

100. Added ***fast_tearardown*** to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

101. Added `SET_MONITOR_COUNTER` command. See [Section 29.35.4, “SET_MONITOR_COUNTER - Set Monitor Counter,” on page 2195.](#)

102. Added `ARM_MONITOR_COUNTER` command. See [Section 29.35.5, “ARM_MONITOR_COUNTER - ARM Monitor Counter,” on page 2198.](#)

103. Added `Monitor_Counter_Change` Event. See [Section 8.20.12.9, “Monitor Counter Change Event,” on page 499.](#)

104. Added `max_num_of_monitor_counters`, `num_q_monitor_counters` and `num_ppcnt_monitor_counters` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

105. Added `dbr_umem_valid` and `dbr_umem_id` to `XRC_SRQ` Context. See [Table 114, “XRC_SRQ Context Layout,” on page 370.](#)

106. Added `ib_striding_wq` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

107. [Internal] Moved field `ib_striding_wq` in `HCA_CAP` from offset 40h, bit 26, to offset 44h, bit 12. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

108. Added Sub-Functions section. See [Section 14.3, “Sub-Function,” on page 845.](#)

109. Added `sf`, `sf_set_partition`, `log_min_sf_size` and `max_num_sf_partitions` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)

110. Added `QUERY_SF_PARTITIONS` command. See [Section 29.3.8, “QUERY_SF_PARTITIONS - Query Sub-Function Partition Configuration,” on page 1810.](#)

111. Added `SET_SF_PARTITIONS` command. See [Section 29.3.9, “SET_SF_PARTITIONS - Set Sub-Function Partition Configuration,” on page 1812.](#)

112. Added `ALLOC_SF` command. See Section 29.3.10, “`ALLOC_SF` - Allocate Sub-Function,” on page 1814.
113. Added `DEALLOC_SF` command. See Section 29.3.11, “`DEALLOC_SF` - De-Allocate Sub-Function,” on page 1815.
114. Added `sw_r_roce_src_udp_port` to ROCE Capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732.
115. Added Multi-port vHCA section. See Section 14.4, “Multi-port vHCA,” on page 845.
116. Added `sw_owner_id` to `INIT_HCA` command. See Table 1666, “`INIT_HCA` Input Structure Layout,” on page 1820.
117. Added `sw_owner_id`, `num_vhca_ports`, `native_port_num` and `affiliate_nic_vport_criteria` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
118. Added `affiliation_criteria` and `affiliated_vhca_id` to `MODIFY_NIC_VPORT_CONTEXT` Bitmask. See Table 35, “`MODIFY_NIC_VPORT_CONTEXT` Bitmask,” on page 293.
119. Added `vhca_port_num` to `QUERY_ROCE_ADDRESS`. See Table 2527, “`QUERY_ROCE_ADDRESS` Input Structure Layout,” on page 2172.
120. Added `vhca_port_num` to `SET_ROCE_ADDRESS`. See Table 2534, “`SET_ROCE_ADDRESS` Input Structure Layout,” on page 2175.
121. Renamed `port` to `vhca_port_num` in QP Address Path. See Table 21, “Address Path Structure Layout,” on page 274.
122. Added `ttl_checksum_correction` to Flow Table NIC Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
123. Added `fdb_single_tir_destination` to E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
124. Renamed `QUERY_EC_PARAMS` command to `QUERY_HOST_PARAMS`. See Section 30.28.1, “`QUERY_HOST_PARAMS` - Query Host CPU Parameters,” on page 1755.
125. Renamed `EC_PARAMS_CHANGE` event command to `HOST_PARAMS_CHANGE`. See Section 8.20.14.4, “`ESW_FUNCTIONS_CHANGED` Event,” on page 503.
126. Added MetaData Registers section. See Section 8.18.2.10, “MetaData Registers,” on page 419.
127. Added `METADATA_REG_C_0-METADATA_REG_C_7` to *field* in `SET_ACTION`. See Table 2205, “`SET_ACTION` Input Structure Field Descriptions,” on page 2051.
128. Added `META_DATA_REG_C_0-METADATA_REG_C_7` to *field* in Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
129. Added `METADATA_REG_C_0-METADATA_REG_C_7` to Flow Table Entry Match Set Mist Parameters 2. See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013.
130. Added `dynamic_qp_allocation` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
131. Added a new TLV to `NV_INTERNAL_RESOURCE_CONF`. See Table 520, “`NV_INTERNAL_HAIRPIN_CONF` Layout,” on page 737.
132. Added a new TLV to `NV_INTERNAL_RESOURCE_CAP`. See Table 522, “`NV_INTERNAL_HAIRPIN_CAP` Layout,” on page 738.

133. Added `pci_atomic_mode` to `NV_SW_OFFLOAD_CONFIG`. See [Table 444, “NV_SW_OFFLOAD_CONFIG Layout,”](#) on page 702.
134. Added the following to `NV_SW_OFFLOAD_CAP`. See [Table 446, “NV_SW_OFFLOAD_CAP Layout,”](#) on page 707:
- `pci_atomics_enabled_ext_atomics_enabled`
 - `pci_atomics_enabled_ext_atomics_enabled_nocoherent`
 - `pci_atomics_enabled_ext_atomics_disabled`
 - `pci_atomics_enabled_ext_atomics_enabled_serialized`
 - `pci_atomics_disabled_ext_atomics_enabled`

Draft G:

135. Added `SEND_NVMMF_CC` command. See [Section 29.10.17, “SEND_NVMMF_CC - Send a NVMMF Command Capsule via NVME SQ,”](#) on page 1936.
136. Added `QUERY_NVMMF_CC_RESPONSE` command. See [Section 29.10.18, “QUERY_NVMMF_CC_RESPONSE - Query NVMMF Command Capsule Response,”](#) on page 1938.
137. Added General Object Types. See [Section 8.27, “General Object Types,”](#) on page 539.
138. Added General Object Commands. See [Section 29.23, “General Object Commands,”](#) on page 2079.
139. Added `NVMe_emulation_to_NVMMf` to `offload_type` in QP Context. See [Table 20, “QP Context Field Descriptions,”](#) on page 261.
140. Added Device Emulation Capabilities. See [Section 29.3.4.13, “NVME Device Emulation Capabilities,”](#) on page 1787.
141. Renamed Device Emulation QP NVME MAP to NVME SQ. See [Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,”](#) on page 542.
142. Added `NVMMf_CC` to `offload_type` in `NVME_SQ` Object. See [Table 273, “NVME_SQ Object Fields,”](#) on page 544.
143. Added QPN to `modify_field_select` in `NVME_SQ` Object. See [Table 273, “NVME_SQ Object Fields,”](#) on page 544.
144. Added `nvme_cq_id`, `log_nvme_page_size`, `max_transaction_size`, `mkey_buffer_umem_offset`, `mkey_buffer_umem_id`, `num_of_namespaces`, and `nvme_namespace` to `NVME_SQ` Object. See [Table 273, “NVME_SQ Object Fields,”](#) on page 544.
145. Added `hide_port2_pf` to `NV_GLOBAL_PCI_CONF`. See [Table 430, “NV_GLOBAL_PCI_CONF Layout,”](#) on page 682.
146. Added `hide_port2_pf_supported` in `NV_GLOBAL_PCI_CAP`. See [Table 433, “NV_GLOBAL_PCI_CAP Field Descriptions,”](#) on page 688.
147. Updated description of `image_bank` field in `NV_FPGA_DATA`. See [Table 513, “NV_FPGA_DATA Field Descriptions,”](#) on page 735.

Draft H:

148. Added `header_modify_sw_icm_start_address` and `log_header_modify_sw_icm_size` to Device Memory Capabilities. See [Table 1605, “Device Memory Capabilities Layout,”](#) on page 1785.

149. Added *flow_source* and *esw_fdb_flow_table_root_per_port* fields in Flow Table E-switch Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748.
150. Added support for setting *other_vport* in SET_FLOW_TABLE_ROOT Input Structure for table type E-switch FDB. See Table 2112, “SET_FLOW_TABLE_ROOT Input Structure Layout,” on page 1995,
151. Added *flow_source* field to FLOW_CONTEXT Structure. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.
152. Added *cmd_on_behalf* and *function_id* to DESTROY_MKEY Input. See Table 1811, “DESTROY_MKEY Input Structure Layout,” on page 1868.
153. Added NV_INTERNAL_CPU_CONF TLV. See Section 11.7.10, “NV_INTERNAL_CPU_CONF,” on page 664.
154. Added NV_INTERNAL_CPU_CAP TLV. See Section 11.7.11, “NV_INTERNAL_CPU_CAP,” on page 665.
155. Added NV_HOST_SYNC_CONF TLV. See “NV_HOST_SYNC_CONF” on page 766.
156. Added *lag_disable* to “NV_PACKET_PACING” on page 710.
157. Added NV_ECPF_CONF TLV. See Section 11.7.25, “NV_ECPF_CONF,” on page 679.
158. Added NV_ECPF_CAP TLV. See Section 11.7.26, “NV_ECPF_CAP,” on page 680.
159. Added MMHI - Multi-Host Information register. See Section 28.3.5, “MMHI - Multi-Host Information Register,” on page 1089.

Revision 0.47 (May 2018)

1. Updated counters description in NVMF NAMESPACE CONTEXT. See Table 2446, “NVMF_NAMESPACE_CONTEXT input Structure Field Descriptions,” on page 2139.
2. Added Fast Teardown section. See Section 7.3.1, “Fast Teardown,” on page 245.
3. Added *fast_teardown* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
4. Added *prepare_fast_teardown* profile to TEARDOWN_HCA. See Table 1671, “TEARDOWN_HCA Input Structure Field Descriptions,” on page 1822.
5. Renamed *force_state* to *state* in TEARDOWN_HCA output command. See Table 1673, “TEARDOWN_HCA Output Structure Field Descriptions,” on page 1823.
6. Added *general_obj_in_cmd_hdr* to General Objects commands. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082.
7. Added *general_obj_out_cmd_hdr* to General Objects commands. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082.
8. Updated NV_FLEX_PARS_CONF TLV. See Table 398, “NV_FLEX_PARSER_CONF Layout,” on page 665.
9. Updated NV_FLEX_PARS_CAP TLV. See Table 400, “NV_FLEX_PARSER_CAP Layout,” on page 666.
10. Added SAFE_MODE_CONF. See Table 408, “SAFE_MODE_CONF Layout,” on page 672.
11. Updated *flex_parser_protocols* description in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

12. Added *metadata_reg_a_width* and *metadata_reg_b_width* to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
13. Added *metadata_reg_a* and *metadata_reg_b* to Flow Table Fields Supported. See Table 1592, “Flow Table Fields Supported Fields,” on page 1762.
14. Added *flow_table_metadata* to Eth segment. See Table 47, “Eth Segment Format,” on page 318.
15. Added *flow_table_metadata* to CQE format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435.
16. Added *metadata_reg_a* to Flow Table Entry Match Misc Parameters 2. See Table 2133, “Flow Table Entry Match Set Misc Parameters 2 Fields,” on page 2014.
17. Added *META_DATA_REG_A* and *META_DATA_REG_B* to SET_ACTION. See Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051.
18. Added *max_encap_header_size*, *log_max_encap_headers* and *encap_general_header* to Flow Table NIC Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
19. Added *encap_general_header* to E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
20. Added GENERAL_HEADER to ENCAP HEADER. See Table 2191, “PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions,” on page 2045.
21. Renamed *mpls* fields to *first_mpls* in Flow Table Fields Supported. See Table 1592, “Flow Table Fields Supported Fields,” on page 1762.
22. Added *outer_first_mpls_over_udp_label*, *outer_first_mpls_over_udp_exp*, *outer_first_mpls_over_udp_s_bos* and *outer_first_mpls_over_udp_ttl* fields in Flow Table Fields Supported. See Table 1592, “Flow Table Fields Supported Fields,” on page 1762.
23. Added *outer_first_mpls_over_gre_label*, *outer_first_mpls_over_gre_exp*, *outer_first_mpls_over_gre_s_bos* and *outer_first_mpls_over_gre_ttl* fields in Flow Table Fields Supported. See Table 1592, “Flow Table Fields Supported Fields,” on page 1762.
24. Renamed *outer_mpls* fields to *outer_first_mpls* in Flow Table Entry Match Misc Parameters 2. See Table 2133, “Flow Table Entry Match Set Misc Parameters 2 Fields,” on page 2014.
25. Renamed *inner_mpls* fields to *inner_first_mpls* in Flow Table Entry Match Misc Parameters 2. See Table 2133, “Flow Table Entry Match Set Misc Parameters 2 Fields,” on page 2014.
26. Added *outer_first_mpls_over_gre_label*, *outer_firstmpls_over_gre_exp*, *outer_first_mpls_over_gre_s_bos* and *outer_first_mpls_over_gre_ttl* to Flow Table Entry Match Misc Parameters 2. See Table 2133, “Flow Table Entry Match Set Misc Parameters 2 Fields,” on page 2014.
27. Added *outer_first_mpls_over_udp_label*, *outer_first_mpls_over_udp_exp*, *outer_first_mpls_over_udp_s_bos* and *outer_first_mpls_over_udp_ttl* to Flow Table Entry Match Misc Parameters 2. See Table 2133, “Flow Table Entry Match Set Misc Parameters 2 Fields,” on page 2014.
28. Added *push_vlan_2_tag* to *modify_enable_mask* in Set Flow Table Entry Command. See Table 2147, “SET_FLOW_TABLE_ENTRY - Input Structure Field Descriptions,” on page 2023.
29. Added *push_vlan_2_tag* to Flow Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.

30. Added *pop_vlan_2* and *push_vlan_2* to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
31. Added **POP_VLAN_2** and **PUSH_VLAN_2** actions to Flow Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.
32. Added Multi vHCA Mkey section, See Section 14.8, “Multi vHCA Mkey,” on page 856
33. Added Multi PCI Port section, See Section 14.9, “Multi PCI Port,” on page 857
34. Added *m_vhca_mk* and *m_pci_port* to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
35. Added *m_pci_port_child*, *m_vhca_mk_child* and *parent_vhca_id* to INIT_HCA command, See Table 1666, “INIT_HCA Input Structure Layout,” on page 1820
36. Added *inline_rqn_vhca_id_valid* and *inline_rqn_vhca_id* to TIR context, See Table 116, “TIR Context Format,” on page 374
37. Added *rq_vhca_id_valid* and *rq_vhca_id* list to RQT context, See Table 130, “RQT Context Format,” on page 399
38. Added *qpn_vhca_id_valid* and *qpn_vhca_id* to ATTACH_TO_MCG, See Table 2619, “ATTACH_TO_MCG Input Structure Layout,” on page 2214
39. Added QUERY_DRIVER_VERSION command. See Section 29.3.20, “QUERY_DRIVER_VERSION,” on page 1830
40. Added *query_driver_version* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
41. Added *max_qp_retry_freq* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
42. Added *max_qp_retry_limit* and *max_qp_retry_period* to Nic Vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
43. Added *max_qp_retry_limit* and *max_qp_retry_period* to Modify Nic Vport Context. See Table 35, “MODIFY_NIC_VPORT_CONTEXT Bitmask,” on page 293.
44. Added *max_qp_retry_freq_exceeded* to QUERY_Q_Counter. See Table 2562, “QUERY_Q_COUNTER Output Structure Layout,” on page 2188
45. Added *traffic_fast_control* to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
46. Added *traffic_state* to INIT_SEGMENT, See
47. Added *ipoib_enhanced_pkey_change* to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
48. Added *cmd_on_behalf* to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
49. Added *cmd_on_behalf* and *cmd_on_behalf_vhca_id* to INIT_HCA, See Table 1666, “INIT_HCA Input Structure Layout,” on page 1820
50. Added *cmd_on_behalf* and *cmd_on_behalf_vhca_id* to ENABLE_HCA, See Table 1674, “ENABLE_HCA Input Structure Layout,” on page 1823
51. Added *cmd_on_behalf* and *cmd_on_behalf_vhca_id* to DISABLE_HCA, See Table 1678, “DISABLE_HCA Input Structure Layout,” on page 1825
52. Added *cmd_on_behalf* and *cmd_on_behalf_vhca_id* to QP Commands, See Section 29.10, “QP Commands,” on page 1897

53. Added `wq_umem_offset` and `wq_umem_id` to `CREATE_QP` command, See Table 1881, “`CREATE_QP` Input Structure Layout,” on page 1899
54. Added `dbr_umem_id` to QP Context, See Table 19, “QP Context Structure Layout,” on page 258
55. Added `dbr_umem_id` to CQ Context, See Table 171, “Completion Queue Context Layout,” on page 467
56. Added `dbr_umem_id`, `wq_umem_id` and `wq_umem_offset` to WQ Context, See
57. Added `cq_umem_id` and `cq_umem_offset` to `CREATE_CQ`, See Table 1860, “`CREATE_CQ` Input Structure Layout,” on page 1888
58. Added `cq_umem_id` and `cq_umem_offset` to `CREATE_MKEY`, See Table 1797, “`CREATE_MKEY` Input Structure Layout,” on page 1861
59. Added `rts2rts_qp_counters_set_id` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
60. Added `inner_first_mpls_ttl` to Flow Table Entry Match Set Parameters 2. See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013
61. Added `cmd_on_behalf` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671
62. Replaced `cmd_on_behalf_vhca_id` with `function_id` `INIT_HCA`, See Table 1666, “`INIT_HCA` Input Structure Layout,” on page 1820
63. Removed `cmd_on_behalf_vhca_id` to `ENABLE_HCA`, See Table 1674, “`ENABLE_HCA` Input Structure Layout,” on page 1823
64. Removed `cmd_on_behalf_vhca_id` to `DISABLE_HCA`, See Table 1678, “`DISABLE_HCA` Input Structure Layout,” on page 1825
65. Replaced `cmd_on_behalf_vhca_id` with `function_id` in QP Commands, See Section 29.10, “QP Commands,” on page 1897
66. Added `device_emulation_manager` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
67. Added Device Emulation Capabilities, See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787
68. Added Device Emulation Event, See Section 8.27.3, “NVMe Device Emulation Object,” on page 541
69. Added Device Emulation Bar Registers Object, See Section 8.27.3, “NVMe Device Emulation Object,” on page 541
70. Added Device Emulation QP to NVMe MAP Object, See Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542
71. Added Mini CQE with Striding RQ section. See Section 8.19.9.6.2, “Mini CQE with Striding RQ,” on page 461.
72. Added `mini_cqe_resp_stride_index` to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
73. Added `resp_bytcnt_checksum_stride_index` to `mini_cqe_res_format` in CQ Context. See Table 172, “Completion Queue Context Field Descriptions,” on page 468.
74. Added Responder Mini CQE With Checksum and Stride Index format. See Table 167, “Responder Mini CQE With Checksum and Stride Index Layout,” on page 463.

75. Added *terminate_scatter_list_mkey* and *repeated_mkey* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
76. Added *terminate_scatter_list_mkey* and *repeated_mkey* to QUERY_SPECIAL_CONTEXT. See Table 1819, “QUERY_SPECIAL_CONTEXTS Input Structure Layout,” on page 1871.
77. Replaced const_0x400 with *repeated_block_mkey* in repeated block format. See Table 706, “Repeated Block Format,” on page 919.
78. Removed snapshot from HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
79. Removed Snapshot Capability from QEURY_HCA_CAP command, See Table 1568, “QEURY_HCA_CAP Input Structure Field Descriptions,” on page 1667
80. Removed Snapshot Capability from SET_HCA_CAP command, See Table 1628, “SET_HCA_CAP Input Structure Field Descriptions,” on page 1804
81. Removed Pre_INIT from op_mod in INIT_HCA command, See Table 1667, “INIT_HCA Input Structure Field Descriptions,” on page 1820
82. Removed suspended from op_mod in CREATE_QP command, See Table 1882, “CREATE_QP Input Structure Field Descriptions,” on page 1900
83. Added qp_by_name to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
84. Added mkey_by_name to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
85. Added *create_qp_start_hint* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
86. Added *encap_and_vlan_action* and *modify_and_vlan_action* to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
87. Added *multi_fdb_encap* to Flow Table E-switch Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748.
88. Renamed rq_vhca_id_format to rq_vhca_id_format in RQT Context, See Table 130, “RQT Context Format,” on page 399
89. Added RQ List. See Table 132, “RQ List Format,” on page 401.
90. Added RQ vHCA List ,See Table 134, “RQ vHCA List Format,” on page 401
91. Added *mkey_by_name* to NV_SW_OFFLOAD_CONFIG. See Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702.
92. Added *mkey_by_name_supported* to NV_SW_OFFLOAD_CAP. See Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707.
93. Added mlxcfg_name for *undi_network_wait*. See Table 592, “NV_ROM_BOOT_CONF2 Layout,” on page 784.

Revision 0.46 (March 2018)

1. Added *initiator_src_dct* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
2. Added *initiator_src_dct* to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
3. Added *initiator_src_dct_in_cqe* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671

4. Added *initiator_src_dct* to CQ Context. See Table 171, “Completion Queue Context Layout,” on page 467.
5. Added *initiator_src_dct* to CQE format. See Table 146, “128B Completion Queue Entry Format Layout,” on page 436.
6. Added *initiator_src_dct* to INIT2RTR required parameters, See Table 1904, “INIT2RTR_QP Required Parameters,” on page 1911.
7. Added *fpga_vendor_acceleration* to Flow Table Properties, See See Table 1587, “Flow Table Properties Layout,” on page 1753
8. Added *fpga_vendor_acceleration* to action field in Flow Context, See See Table 2149, “FLOW_CONTEXT Field Descriptions,” on page 2026
9. Added *mpls* capabilities to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
10. Added *misc_parameters_2* to *match_criteria_enable* in CREATE_FLOW_GROUP command. See Table 2121, “CREATE_FLOW_GROUP - Input Structure Field Description,” on page 2001.
11. Added Flow Table Entry Match Set Parameters 2. See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013.
12. Added *embedded_cpu_function* QUERY_PAGES Input Structure, See Table 1553, “QUERY_PAGES Input Structure Layout,” on page 1660.
13. Added *log_min_hairpin_wq_data_sz* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
14. Added *Factory_failover* to *flash_select_oper* in FPGA_CTRL, See Table 896, “FPGA_CTRL Register Fields,” on page 1110
15. Added *merged_eswitch* to E-switch Capabilities, See Table 1593, “E-switch Capabilities Layout,” on page 1768
16. Added *Destination_eswitch_owner_vhca_id_valid* and *Destination_eswitch_owner_vhca_id* to Destination format, See Table 2158, “Destination Entry Format Structure Layout,” on page 2031.
17. Added *source_eswitch_owner_vhca_id_valid* to CREATE_FLOW_GROUP command, See Table 2120, “CREATE_FLOW_GROUP Input Structure Layout,” on page 2000
18. Added *source_eswitch_owner_vhca_id* to Flow Table Entry Match Set Misc Parameters, See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009
19. Added *source_vhca_port* to Flow Table Fields Supported, See Table 1591, “Flow Table Fields Supported Format,” on page 1761
20. Added *source_vhca_port* to Flow Table Entry Match Set Misc Parameters, See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009
21. Changed *fpga_id* of Newton_X to 0x0 instead of 0x1, See Table 892, “FPGA_CAP Register Fields,” on page 1105
22. Added *fpga_id* of Edison, See Table 892, “FPGA_CAP Register Fields,” on page 1105
23. Added *afu_owner* to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
24. Added AFU commands, See Section 29.21, “AFU Command - Acceleration Functions Units commands,” on page 2070
25. Added *afu_id* to SQ Context, See Table 138, “SQ Context Format,” on page 404

26. Added *afu_id* to TIR Context, See Table 116, “TIR Context Format,” on page 374
27. Added Ethernet Visualization in BlueField devices section, See Section 14.5, “Ethernet Virtualization in CPU Devices (BlueField),” on page 846
28. Added *embedded_cpu_function* to ENABLE_HCA command. See Section 29.3.15, “ENABLE_HCA,” on page 1823.
29. Added *embedded_cpu_function* to DISABLE_HCA command. See Section 29.3.16, “DISABLE_HCA,” on page 1825
30. Updated *embedded_cpu* to Initialization Segment. See Table 14, “Initialization Segment,” on page 234.
31. Updated *embedded_cpu_function* to QUERY_PAGES command. See Table 1553, “QUERY_PAGES Input Structure Layout,” on page 1660.
32. Updated *embedded_cpu_function* to MANAGE_PAGES command. See Table 1557, “MANAGE_PAGES Input Structure Layout,” on page 1663.
33. Updated *embedded_cpu_function* to Pages Request event. See Table 234, “Pages Request Event Layout,” on page 502.
34. Updated QUERY_EC_PARAMS command. See Section 30.28.1, “QUERY_HOST_PARAMS - Query Host CPU Parameters,” on page 1755.
35. Updated EC_PARAMS_CHANGE event. See Section 8.20.14.4, “ESW_FUNCTIONS_CHANGED Event,” on page 503.
36. Added merged_eswitch to E-switch capabilities, See Table 1593, “E-switch Capabilities Layout,” on page 1768.
37. Added *counter_eswitch_affinity* to E-switch capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.
38. Added Resource management for vNICs section. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.
39. Added *other_function* and *function_id* to QUERY_HCA_CAP. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667.
40. Added *other_function* and *function_id* to SET_HCA_CAP. See Section 29.3.5, “SET_HCA_CAP – Set Device Capabilities,” on page 1803.
41. Added *vhca_resource_manager* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
42. Added a “Modifiable by *vhca_resource_manager*” column to HCA_CAP field description table. See Table 1572, “HCA Capabilities Field Descriptions,” on page 1679.
43. Removed *port_num* from QUERY_VNIC_ENV command. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181.
44. Added *nic_receive_steering_discard*, *receive_discard_vport_down*, *transmit_discard_vport_down* and *eq_overrun_count* to HCA_CAP. See Table 1572, “HCA Capabilities Field Descriptions,” on page 1679.
45. Added *nic_receive_steering_discard*, *receive_discard_vport_down*, *transmit_discard_vport_down*, *async_eq_overrun* and *comp_eq_overrun* to QUERY_VNIC_ENV. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181.

46. Added Device Health Events section. See [Section 27.6, “Device Health Events,”](#) on page 1061.
47. Added Receive Hang and Transmit Engine Hang sub events to General notification event. See [Table 215, “General Notification Event Subtype,”](#) on page 496.
48. Added `uefi_undi_vlan_config_en` to `NV_ROM_CAP`. See [Table 590, “NV_ROM_CAP Layout,”](#) on page 782
49. Added `uefi_undi_vlan_en` and `uefi_undi_vlan_id` to `NV_ROM_UEFI_CONF`, See [Table 594, “NV_ROM_UEFI_CONF Layout,”](#) on page 784.
50. Added `vlan_stripping_supported` to NODNIC Configuration Registers. See [Table 660, “NODNIC Configuration Registers Layout,”](#) on page 815.
51. Added `vlan_stripping_en` to NODNIC Port Configuration. See [Table 662, “NODNIC Port Configuration Registers Layout,”](#) on page 817.
52. Updated Striding RQ section. See [Section 8.9.1.3, “Striding Receive WQ,”](#) on page 304.
53. Renamed `wqe_id` to `eth_wqe_id` in CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
54. Updated description of `eth_wqe_id`, `byte_cnt` and `wqe_counter` in CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
55. Added `ib_stride_index` to CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
56. Added “Modifiable by `vhca_resource_manager`” column to Flow Table Nic Capabilities. See [Table 1584, “Flow Table NIC Capabilities Field Descriptions,”](#) on page 1745.
57. Added `nic_receive_max_steering_depth` to Flow Table Nic Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.
58. Added `nic_receive_steering_depth` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
59. Added `nic_receive_steering_depth` to `QUERY_VNIC_ENV`. See [Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,”](#) on page 2181.
60. Added `fdb_multi_path_to_table`, `egress_acl_forward_to_vport` and `ingress_acl_forward_to_vport` to E-switch Capabilities. See [Table 1585, “Flow Table E-switch Capabilities Layout,”](#) on page 1748.
61. Updated Ingress ACL section to support forward to Vport action. See [Table 8.18.4.8, “E-Switch Ingress Access Control List \(ACL\),”](#) on page 431.
62. Updated E-switch Forwarding section to support multi-path. See [Table 8.18.4.7, “E-Switch Forwarding Data Base \(FDB\),”](#) on page 430.
63. Added `invalid_command_count` and `quota_exceeded_count` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
64. Added `invalid_command` and `quota_exceeded_command` to `QUERY_VNIC_ENV`. See [Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,”](#) on page 2181.
65. Renamed `panic_state` to `force_state` in `TEARDOWN_HCA` output structure. See [Table 1672, “TEARDOWN_HCA Output Structure Layout,”](#) on page 1823. Added Core Dump section, See [Section 27.9, “Core Dump,”](#) on page 1078
66. Added `core_dump_general` and `core_dump_qp` to Debug Capabilities, See [Table 1599, “Debug Capabilities Layout,”](#) on page 1777

67. Added `CORE_DUMP` register, See Section 28.11, “CORE DUMP Register,” on page 1128
68. Moved `log_response_bar_size` and `response_bar_address` in `SET_TUNNELED_OPERATION_PARAMS` command to the output of the command, See Table 1563, “SET_TUNNELED_OPERATION_PARAMS Input Structure Layout,” on page 1665
69. Renamed `disable_local_lb` to `disable_local_lb_uc` in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
70. Added `disable_local_lb_mc` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
71. Added `pci_data_wr_ordering_supported` to NV_HCA_CAP. See Table 422, “NV_HCA_CAP Layout,” on page 679.
72. Added `pci_data_wr_ordering_mode` to NV_HCA_CONF. See Table 420, “NV_HCA_CONF Layout,” on page 678.
73. Added `umr_indirect_mkey_disabled`, `umr_modify_stomic_disabled`, `umr_modify_entity_size_disabled` and `repeated_block_disabled` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
74. Added `trusted_vnic_vhca` to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
75. Added Driver Tracer Event. See Section 8.20.11.2, “Driver Tracer Event,” on page 488
76. Renamed `striding_rq` to `eth_striding_wq` in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
77. Renamed `ib_striding_rq` to `ib_striding_wq` in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
78. Added `log_max_cmd_timeout` and `log_min_cmd_timeout` to NVMe-oF capabilities. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.
79. Added `log_cmd_timeout_us` to NVMF BACKEND CONTROLLER CONTEXT. See Table 2423, “NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130.
80. Removed `offloaded_capsules_count` from NVMe-oF Target Offload Context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
81. Added `cmd_unknown_namespace_cnt` to NVMe-oF Target Offload Context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
82. Added `cmd_unknown_namespace_cnt` and `frontend_namespace_context` to NVMe-oF capabilities. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.
83. Added `max_memic_addr_alignment` to MEMIC capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
84. Added `memic_addr_alignment` to `ALLOC_MEMIC`. See Section 29.6.6, “ALLOC_MEMIC – Allocate MEMIC,” on page 1873.
85. Added `QUERY_NVMF_NAMESPACE_CONTEXT` command. See Section 29.28.6, “QUERY_NVMF_NAMESPACE_CONTEXT – Query NVMF NameSpace Context,” on page 2137
86. Added `rfr` to Health Buffer. See Table 813, “Health Buffer Layout,” on page 1049.

87. Added `reset_with_host_on_errors` and `sw_recovery_on_error` to `NV_POWER_CONF`, See Table 562, “`NV_POWER_CONF` Layout,” on page 763
88. Added `reset_with_host_on_errors_supported` and `sw_recovery_on_error_supported` to `NV_POWER_CAP`, See Table 564, “`NV_POWER_CAP` Layout,” on page 765
89. Added `ibm_as_notify_en` and `ibm_tunneled_atomic_en` to `NV_PCI_CONF`, See Table 404, “`NV_PCI_CONF` Layout,” on page 668
90. Added `ibm_as_notify_supported` and `ibm_tunneled_atomic_supported` to `NV_PCI_CAP`, See Table 406, “`NV_PCI_CAP` Layout,” on page 670
91. Added `NV_SECURE_HOST` TLV, see Section 11.7.65, “`NV_SECURE_HOST`,” on page 731
92. Added `FILE_SIGNATURE_4096`, See Section 11.7.87, “`FILE_SIGNATURE_4096_A`,” on page 752
93. Added `FILE_PUBLIC_KEY_4096`, See Section 11.7.90, “`FILE_PUBLIC_KEY_4096_A`,” on page 755
94. Added `NV_INTERNAL_CPU_CONF` TLV, See Section 11.7.10, “`NV_INTERNAL_CPU_CONF`,” on page 664
95. Added `NV_INTERNAL_CPU_CAP` TLV, See Section 11.7.11, “`NV_INTERNAL_CPU_CAP`,” on page 665
96. Added `NV_ECPF_CONF` TLV, See Section 11.7.25, “`NV_ECPF_CONF`,” on page 679
97. Added `NV_ECPF_CAP` TLV, See Section 11.7.26, “`NV_ECPF_CAP`,” on page 680
98. Added `SECURE_HOST` register. See Section 28.10, “`SECURE_HOST`,” on page 1127.
99. Updated `max_trace_buffer_size` to `log_max_trace_buffer_size` and fixed size to bits 7:0. See Table 1478, “`MTRC_CAP` - Tracer Capability Register Layout,” on page 1587.
100. Updated `trace_buffer_size` to `log_trace_buffer_size` and fixed size to bits 7:0. See Table 1482, “`MTRC_CONF` - Tracer Configuration Register Layout,” on page 1589.
101. Added `NV_HOST_BOOT_PREP_CONF` TLV. See Section 11.4.1, “`NV_HOST_BOOT_PREP_CONF`,” on page 519.
102. Added `NV_HOST_BOOT_PREP_CAP` TLV. See Section 11.4.2, “`NV_HOST_BOOT_PREP_CAP`,” on page 520.
103. Moved *`extended_retry_count_valid`* from bit 19 to bit 12 to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
104. Added FW stressor registers. See Chapter 28, “Command Registers” (page 1080).
105. Added Stress Generation section. See Section 27.10, “Stress Generation,” on page 1078.
106. Added *`strs_cap`* to `ICMD_QUERY_Capability`. See Section 30.4.1, “`ICMD_QUERY_CAPABILITY` - Get ICMD Interface Capabilities,” on page 2273.
107. Renamed *`vnic_env`* to *`vnic_env_queue_counters`* in `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
108. Added *`packet_pacing_typical_size`* to QoS Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
109. Added *`typical_packet_size`* to `SET_PP_RATE_LIMIT`. See Table 2574, “`SET_PP_RATE_LIMIT` Input Structure Layout,” on page 2199.
110. Added *`typical_packet_size`* to `QUERY_PP_RATE_LIMIT`. See Table 2580, “`QUERY_PP_RATE_LIMIT` Input Structure Layout,” on page 2201.

- 111.Updated Packet Pacing section. See [Section •](#), “ETS TSAR (root) followed by a sequence of DWRR or Round Robin TSARs, so the path (excluding the leaf) is shorter or equal to $2^{HCA\_CAP.log\_esw\_max\_sched\_depth}$ but holds at least one TSAR.” on page 928.
- 112.Replaced FILE_SIGNATURE_4096 TLV with FILE_SIGNATURE_4096_A + FILE_SIGNATURE_4096_B
- 113.Replaced FILE_PUBLIC_KEY_4096 with FILE_PUBLIC_KEY_4096_A + FILE_PUBLIC_KEY_4096_B
- 114.Deleted NV_HOST_BOOT_PREP_CONF and NV_HOST_BOOT_PREP_CAP
- 115.Added exp_rom_pxe_supported, exp_rom_uefi_x86_supported, exp_rom_uefi_arm_supported to NV_PCI_CAP
- 116.Added exp_rom_pxe_enable, exp_rom_uefi_x86_enable, exp_rom_uefi_arm_enable to NV_PCI_CONF
- 117.Added NV_FLEX_PARS_CONF TLV, See Table 398, “NV_FLEX_PARSER_CONF Layout,” on page 665
- 118.Added NV_FLEX_PARS_CAP TLV, See Table 400, “NV_FLEX_PARSER_CAP Layout,” on page 666
- 119.Added flex_parser_protocols to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
- 120.Added MODIFY_NIC_VPORT_CONTEXT Bitmask. See [Table 35](#), “MODIFY_NIC_VPORT_CONTEXT Bitmask,” on page 293.
- 121.Updated description of *other_vport* in Modify Vport state command. See [Section 29.31.2](#), “MODIFY_VPORT_STATE – Modify Vport State,” on page 2157.
- 122.Renamed *eswitch_flow_table* to *eswitch_manager* in HCA_CAP. See [Table 1571](#), “HCA Capabilities Layout,” on page 1671.
- 123.Added *rts2rts_qp_dscp* to HCA_CAP. See [Table 1571](#), “HCA Capabilities Layout,” on page 1671.
- 124.Added *dscp* and *f_dscp* to RTS2RTS Optional parameters. See [Table 1916](#), “RTS2RTS_QP Optional Parameters Mask,” on page 1919.
- 125.Renamed *current_error_queues* to *current_q_under_processor_handle* in vNIC Diagnostic Statistics. See [Table 2550](#), “vNIC Diagnostic Statistics Layout,” on page 2183.
- 126.Renamed *total_error_queues* to *total_q_under_processor_handle* in vNIC Diagnostic Statistics. See [Table 2550](#), “vNIC Diagnostic Statistics Layout,” on page 2183.
- 127.Added GENERAL OBJECTS commands, See [Section 29.23](#), “General Object Commands,” on page 2079
- 128.Added general_obj_types to HCA_CAP, See [Table 1571](#), “HCA Capabilities Layout,” on page 1671
- 129.Added General Object Types section, See [Section 8.27](#), “General Object Types,” on page 539
- 130.Added UID to all commands. See [Chapter 29](#), “Command Reference” (page 1647)

Revision 0.45 (October 2017)

1. Added *nvmf_target_offload* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
2. Added *log_max_backend_controller_per_xrq* to NVME Capabilities. See [Table 1603, “NVMe-oF Capabilities Layout,” on page 1780](#).
3. Renamed *panic_tearardown* to *force_tearardown* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
4. Renamed *panic_close* to *force_close* in TEARDOWN command. See [Table 1671, “TEARDOWN_HCA Input Structure Field Descriptions,” on page 1822](#).
5. Renamed *log_max_hairpin_wq_sz* to *log_max_hairpin_num_packets* in HCA_CAP, See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
6. Added *log_max_hairpin_wq_data_sz* and *log_max_hairpin_queues* to CA_CAP, See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
7. Added *log_hairpin_num_packets* and *log_hairpin_data_sz* to WQ context, See [Table 127, “Work Queue \(WQ\) Fields,” on page 391](#).
8. Added hairpin, *hairpin_peer_vhca* and *hairpin_peer_rq* to *CREATE* and *MODIFY SQ bitmask*, See [Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407](#).
9. Added hairpin, *hairpin_peer_vhca* and *hairpin_peer_sq* to *CREATE* and *MODIFY RQ bitmask*, See [Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395](#).
10. Added *log_max_dc_cnak_qps* to HCA_CAP, See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
11. Renamed *max_flow_counter* to *max_flow_counter_15_0* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
12. Added *max_flow_counter_31_16* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
13. Resized *flow_counter_id* from 16 bits to 32 bits in Flow Counter List Entry. See [Table 2160, “Flow Counter List Entry Format Structure Layout,” on page 2032](#).
14. Resized *flow_counter_id* from 16 bits to 32 bits in ALLOC FLOW COUNTER command. See [Table 2176, “ALLOC_FLOW_COUNTER Output Structure Layout,” on page 2039](#).
15. Resized *flow_counter_id* from 16 bits to 32 bits in DEALLOC FLOW COUNTER command. See [Table 2178, “DEALLOC_FLOW_COUNTER Input Structure Layout,” on page 2040](#).
16. Resized *flow_counter_id* from 16 bits to 32 bits in QUERY FLOW COUNTER command. See [Table 2182, “QUERY_FLOW_COUNTER Input Structure Layout,” on page 2041](#).
17. Modified *traffic_class* place in PARA_VPORT_TC Element, See [Table 2594, “PARA_VPORT_TC Element Attributes Format,” on page 2206](#).
18. Added *outer_emd_tag* to Flow Table Fields Supported, See [Table 1591, “Flow Table Fields Supported Format,” on page 1761](#).
19. Added *outer_emd_tag* and *outer_emd_tag_data[0..5]* to Flow Table Entry Match Set Misc, See [Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009](#).
20. Added *FLASH_GW_LOCK* and *FLASH_GW_UNLOCK* to operation in FPGA_CTRL, See [Table 896, “FPGA_CTRL Register Fields,” on page 1110](#).

21. Added *cqe_compression_128b* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
22. Added Multi-port vHCA section, See [Section 14.4, “Multi-port vHCA,” on page 845](#)
23. Added *sw_owner_id* to INIT_HCA command, See [Table 1666, “INIT_HCA Input Structure Layout,” on page 1820](#)
24. Added *sw_owner_id*, *num_vhca_ports*, *native_port_num* and *affiliate_nic_vport_criteria* to HCA_CAP, See [Table 1571, “HCA Capabilities Layout,” on page 1671](#)
25. Added *affiliation_criteria* and *affiliated_vhca_id* to Nic Vport Context, See [Table 30, “NIC_Vport Context Field Descriptions,” on page 289](#)
26. Added *affiliation* to *field_select* in MODIFY_NIC_VPORT_CONTEXT, See [Section 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164](#)
27. Added *vhca_port_num* to QUERY_ROCE_ADDRESS, See [Table 2527, “QUERY_ROCE_ADDRESS Input Structure Layout,” on page 2172](#)
28. Added *vhca_port_num* to SET_ROCE_ADDRESS, See [Table 2534, “SET_ROCE_ADDRESS Input Structure Layout,” on page 2175](#)
29. Renamed *port* to *vhca_port_num* in QP Address Path, See [Table 21, “Address Path Structure Layout,” on page 274](#)
30. Added QUERY_XRQ_DC_PARAMS_ENTRY command. See [Section 29.27.5, “QUERY_XRQ_DC_PARAMS_ENTRY – Query XRQ DC PARAMS Entry,” on page 2122.](#)
31. Added SET_XRQ_DC_PARAMS_ENTRY command. See [Section 29.27.6, “SET_XRQ_DC_PARAMS_ENTRY – SET XRQ DC PARAMS Entry,” on page 2124.](#)
32. Removed POST_SEND_OFFLOADED_QP command.
33. Removed OFFLOAD_ERROR event.
- 34.
35. Added XRQ_ERROR_EVENT. See [Section 8.20.10, “XRQ Error Event,” on page 482.](#)
36. Added QUERY_XRQ_ERROR_PARAMS command. See [Section 29.27.7, “QUERY_XRQ_ERROR_PARAMS – Query XRQ Error Parameters,” on page 2125.](#) Added *cqe_128_always* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
37. Added *BYTES_128_ALWAYS* to *cqe_sz* in CQ context. See [Table 171, “Completion Queue Context Layout,” on page 467.](#)
38. Added *flash_gw_lock* and *disconnect_fpga* to FPGA_CAP, See [Table 891, “FPGA_CAP Register Layout,” on page 1103](#)
39. Added DISCONNECT and CONNECT operations to FPGA_CTRL, See [Table 896, “FPGA_CTRL Register Fields,” on page 1110](#)
40. Added DISCONNECTED to status in FPGA_CTRL, See [Table 896, “FPGA_CTRL Register Fields,” on page 1110](#)
41. Renamed *dc* to *offloaded_dc_enable* in XRQ context. See [Table 246, “XRQ Context Layout,” on page 510.](#)
42. Added *capi_invalidate* to HCA_CAP, See [Table 1571, “HCA Capabilities Layout,” on page 1671](#)
43. Added *capi_invalidate* to INIT_SEGMNET, See

44. Marked `log2_hds_buf_size` as not supported for XRQ in WQ Context, See Table 127, “Work Queue (WQ) Fields,” on page 391
45. Added ***basic_cyclic_rcv_wqe*** to XRQ Context. See Table 246, “XRQ Context Layout,” on page 510. Added Ethernet Virtualization with Embedded CPU section, See Section 14.5, “Ethernet Virtualization in CPU Devices (BlueField),” on page 846
46. Added ***embedded_cpu*** to Initialization Segment. See Table 14, “Initialization Segment,” on page 234.
47. Added `embedded_cpu_function` to QUERY_PAGES command, See Table 1553, “QUERY_PAGES Input Structure Layout,” on page 1660
48. Added `embedded_cpu_function` to MANAGE_PAGES command, See Table 1557, “MANAGE_PAGES Input Structure Layout,” on page 1663
49. Added `embedded_cpu_function` to Pages Request event, See Table 234, “Pages Request Event Layout,” on page 502
50. Added QUERY_EC_PARAMS command, See Section 30.28.1, “QUERY_HOST_PARAMS - Query Host CPU Parameters,” on page 1755
51. Added EC_PARAMS_CHANGE event, See Section 8.20.14.4, “ESW_FUNCTIONS_CHANGED Event,” on page 503.
52. Added Tunneled Operations section, See Section 23, “Tunneled Operations,” on page 1037
53. Added `as_notify` to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671
54. Added `as_notify`, `process_id`, `thread_id` and `local_partition_id` to CQ context, See Table 171, “Completion Queue Context Layout,” on page 467
55. Added SET_TUNNELED_OPERATION_PARAMS command, See Section 29.3.3, “SET_TUNNELED_OPERATION_PARAMS - Set Tunneled Operation Parameters,” on page 1665
56. Added SET_TUNNELED_OPERATION_PARAMS to Initialization flow, See Section 7.2, “HCA Driver Start-up,” on page 244
57. Added IB routing modes NV cfg. for MPFS
 - Added “NV_MPFS_CAP” on page 674, and the field ***ib_routing_mode_supported***.
 - Added ***ib_routing_mode*** to “NV_MPFS_CONF” on page 673
58. Added IB routing mode NV cfg. for eSwitch
 - Added ***sriov_routing_mode_supported*** in “NV_IB_DC_CAP” on page 641
 - Added ***sriov_routing_mode*** field in “NV_IB_DC_CONF” on page 640
59. Added Push VLAN and Pop VLAN actions to packet steering (#1124821)
 - Added actions definition in “Processing Actions” on page 416
 - Updated ambiguity clarification requirement in “Packet Classification Ambiguities” on page 418
 - Added exclusivity conditions for the actions in “Flow Actions” on page 424
 - Added support limitations for the different Flow Table types in “Characteristics of Flow Table Types” on page 428
 - Added POP_VLAN and PUSH_VLAN values for ***action*** field, and a new ***push_vlan_tag*** field in Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025

- Added capability fields *pop_vlan* and *push_vlan* in Table 1587, “Flow Table Properties Layout,” on page 1753
 - Defined *modify_enable_mask* bit for modifying *push_vlan_tag* in Table 2147, “SET_FLOW_TABLE_ENTRY - Input Structure Field Descriptions,” on page 2023. See also “Redefining a Flow” on page 426
60. Added support for WQEs with less number of strides for Striding RQs (#1085509)
- Added capability field *ext_stride_num_range* in Table 1571, “HCA Capabilities Layout,” on page 1671
 - Increased *single_wqe_log_num_of_strides* field width to 4 bit in Table 126, “Work Queue (WQ) Format,” on page 389 and added supported values -6..-1
61. Added NV configuration for host chaining (#1119999) [Internal]
- Added capabilities TLV “NV_HOST_CHAINING_CAP” on page 662
 - Added configuration TLV “NV_HOST_CHAINING_CONF” on page 661
62. Added NV configuration to disable boot from a port (#1120302) [Internal]
- Added capability field *uefi_undi_config_en* in “NV_ROM_CAP” on page 782
 - Added configuration TLV “NV_ROM_UEFI_CONF” on page 784 with a new field *uefi_undi_dis*.
63. Added NV configuration to disable pre-boot HII menus (#1120307) [Internal]
- Added capability field *hii_config_en* in “NV_ROM_CAP” on page 782
 - Added configuration field *hii_en* in “NV_ROM_UEFI_CONF” on page 784
64. Added NV configuration to disable Expansion ROM LACP (#1113442) [Internal]
- Added capability field *lACP_config_en* in “NV_ROM_CAP” on page 782
 - Added configuration field *lACP_dis* in “NV_ROM_BOOT_CONF1” on page 786
65. Added NV configuration to reload FPGA as part of the device boot (#1119994) [Internal]
- Added capabilities TLV “NV_FPGA_CAP” on page 664 with field *auto_reload_en*.
 - Added configuration TLV “NV_FPGA_CONF” on page 663 with field *auto_reload*.
66. Added NV configuration to enable Multi-port vHCA (#1118487) [Internal]
- Added capabilities TLV “NV_HCA_CAP” on page 679 with field *multi_port_vhca_supported*.
 - Added configuration TLV “NV_HCA_CONF” on page 678 with field *multi_port_vhca_en*.
67. Added NV configuration to enable non-standard IPoVXLAN (#1132312) [Internal]
- Added capabilities TLV “NV_SW_OFFLOAD_CAP” on page 707 with field *ip_over_vxlan_supported*.
 - Added *ip_over_vxlan_en* and *ip_over_vxlan_port* fields in TLV “NV_SW_OFFLOAD_CONFIG” on page 702

68. Added limitation for Flow Tag and Counter in the same FTE (#1040226)
 - Added limitation in “NIC Receive” on page 428
 - Added capability *nic_rx_flow_tag_count_en* in Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742 [Internal Pending implementation]
 69. Fixed enumeration of flash_select_admin and flash_select_oper fields in Table 896, “FPGA_CTRL Register Fields,” on page 1110 (#1125512)
 70. Renamed *tll_hoplimit* to *ip_tll_hoplimit* in Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005
 71. Added Firmware Component Section. See Chapter 10, “Firmware Components Update” (page 635).
 72. Added Section 28.31.1, “MCQS - Management Component Query Status,” on page 1619.
 73. Added Section 28.31.2, “MCQI - Management Component Query Information,” on page 1622.
 74. Added Section 28.31.3, “MCC - Management Component Control,” on page 1629.
 75. Added Section 28.31.4, “MCDA - Management Component Data Access,” on page 1633.
 76. Added Section 28.31.5, “MQIS - Management Query Information Strings,” on page 1634. Added VPort diagnostic monitoring
 - Added description in “vNIC Monitoring” on page 856
 - Added capability field *vnic_env* in “QUERY_HCA_CAP – Query Device Capabilities” on page 1667
 77. Added new command “QUERY_VNIC_ENV – Query vNIC Environment” on page 2181
 78. Added VSEC access syndrome (#930456)
 - Added *syndrome* and *syndrome_code* fields in Table 7, “Vendor Specific Capability Layout,” on page 368
 - Added *syndrome* and *syndrome_code* to the VSEC access flows: “Gateway Read” on page 372 and “Gateway Write” on page 372
 79. Added Space size reporting (#1146384)
 - Added *size_vld* field in Table 7, “Vendor Specific Capability Layout,” on page 368
 - Added description for *address* field in Table 8, “Vendor Specific Capability Field Descriptions,” on page 369
 - Added space parameters query directive in VSEC access flows : “Gateway Read” on page 372 and “Gateway Write” on page 372
 80. Added ND CRSAPCE space (#1146384)
 - Added description in “Nondestructive (ND) CRSPACE Space” on page 370
- Added ND CRSPACE value for *space* field in Table 8, “Vendor Specific Capability Field Descriptions,” on page 369

Revision 0.44 (June 2017)

- Added NVMe capabilities to *op_mod* in QUERY_HCA_CAP. See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667.
- Added NVMe capabilities to *op_mod* in SET_HCA_CAP. See [Table 1628, “SET_HCA_CAP Input Structure Field Descriptions,”](#) on page 1804.
- Added NVMe Capabilities. See [Section 29.3.4.11, “NVMe-oF Capabilities,”](#) on page 1779.
- Removed *nvme_write_offload*, *nvme_read_offload*, *nvme_read_write_offload*, *nvme_read_write_flush_offload*, *log_min_staging_buffer_pages*, *log_max_staging_buffer_pages*, *max_icdoff* and *log_min_ioccsz* from HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added CREATE_NVMEF_BACKEND_CONTROLLER command. See [Section 29.28.1, “CREATE_NVMEF_BACKEND_CONTROLLER – Create NVMe-oF Backend Controller,”](#) on page 2129.
- Added DESTROY_NVMEF_BACKEND_CONTROLLER command. See [Section 29.28.2, “DESTROY_NVMEF_BACKEND_CONTROLLER – Destroy NVMe-oF Backend Controller,”](#) on page 2132.
- Added QUERY_NVMEF_BACKEND_CONTROLLER command. See [Section 29.28.3, “QUERY_NVMEF_BACKEND_CONTROLLER – Query NVMe-oF Backend Controller,”](#) on page 2133.
- Added ATTACH_NVMEF_NAMESPACE command. See [Section 29.28.4, “ATTACH_NVMEF_NAMESPACE – Attach NVMe-oF Namespace,”](#) on page 2134.
- Added DETACH_NVMEF_NAMESPACE command. See [Section 29.28.5, “DETACH_NVMEF_NAMESPACE – Detach NVMe-oF Namespace,”](#) on page 2136.
- Updated NVMe Target Offload Context Header. See [Table 256, “NVMe-oF Target Offload Context Header Layout,”](#) on page 519.
- Renamed *log_max_rndv_payload_size* to *log_max_tm_offloaded_op_size* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Extended *nic_interface* to 3 bits in Initialization Segment. See [Table 14, “Initialization Segment,”](#) on page 234.
- Added *reset* to *nic_interface* in Initialization Segment. See [Table 15, “Initialization Segment Field Descriptions,”](#) on page 235.
- Extended *nic_interface_supported* to 7 bits in Initialization Segment. See [Table 14, “Initialization Segment,”](#) on page 234.
- Added MEMIC section. See [Section 18.4, “MEMIC - Memory Mapped to InterConnect,”](#) on page 923.
- Added Device_Memory capabilities to *op_mod* in QUERY_HCA_CAP, See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667

- Added Device_Memory capabilities to op_mod in SET_HCA_CAP, See Table 1628, “SET_HCA_CAP Input Structure Field Descriptions,” on page 1804
- Added *device_memory* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671
- Added Device Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.
- Added ALLOC_MEMIC command. See Section 29.6.6, “ALLOC_MEMIC – Allocate MEMIC,” on page 1873.
- Added DEALLOC_MEMIC command. See Section 29.6.7, “DEALLOC_MEMIC – De-Allocate MEMIC,” on page 1874.
- Renamed *access_mode* to *access_mode_1_0* in Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- Added *access_mode_4_2* in Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- Added *memic* to *access_mode* in Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- Updated Global Atomicity Guaranteeing section, See Chapter 26, “Global Atomicity Guaranteeing (GAG)” (page 1046).
- Added *pci_atomic_operation* and *pci_atomic_size* to Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728.
- Added *enhanced_retransmission_q_counters* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Enhanced Q Counters to Query Q Counters. See Section 29.35, “Q Counters Commands,” on page 2185.
- Added *relaxed_ordering_write* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *relaxed_ordering_write* to Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- Added *rts2rts_qp_counters_set_id* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *counter_set_id* to RTS2RTS_QP Optional Parameters. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.
- Added *cq_overflow*, *resp_cqe_flush_error* and *req_cqe_flush_error* counters to Q Counters. See Table 2562, “QUERY_Q_COUNTER Output Structure Layout,” on page 2188.
- Updated description of *resp_cqe_error* and *req_cqe_error* in Q Counters. See Table 2562, “QUERY_Q_COUNTER Output Structure Layout,” on page 2188.
- Added *qcam_reg* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Removed *fw_ctrl_update* from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Removed *multi_path* from CREATE_QP Required Parameters. See Table 1885, “CREATE_QP Required Parameters,” on page 1902.
- Added *multi_path* to INIT2RTR_QP Required Parameters. See Table 1904, “INIT2RTR_QP Required Parameters,” on page 1911.
- Added *pcam_reg*, *mcam_reg* and *qcam_reg* to ICMD_QUERY_CAPABILITY. See Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274.
- Removed *hca_sq_owner* from QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Removed *nvmmf_offload_en* from QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *offload_type* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *offload_type* to DCT context. See Table 25, “DCT Context Layout,” on page 279.
- Updated CAPI section. See Chapter 22, “CAPI- Coherent Accelerator Programming Interface” (page 1035).
- Added *capi* to ICMD_QUERY_CAPABILITY. See Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274.
- Added *capi*, *create_pec* and *log_max_pasid* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *ma_translation_mode* to Mkey Context. See Table 362, “MKey Context Format,” on page 629.
- Removed *manage_pages_pasid* and *cmd_pasid* from Initialization Segment. See Table 14, “Initialization Segment,” on page 234.
- Added *pasid* to mkey_ctx_mask in UMR control segment. See Table 53, “UMR Control Segment Fields,” on page 323.
- Added CREATE_CAPI_PEC command. See Section 29.22.1, “CREATE_CAPI_PEC,” on page 2075.
- Added QUERY_CAPI_PEC command. See Section 29.22.2, “QUERY_CAPI_PEC,” on page 2077.
- Added DESTROY_CAPI_PEC command. See Section 29.22.3, “DESTROY_CAPI_PEC,” on page 2078.
- Added CAPI_CTRL register. See Table 939, “CAPI_CTRL Register Layout,” on page 1132.

- Added PPTB and PBMC support. See [Section 28.19, “Network Ports Registers,”](#) on page 1140.
- Added *qpts_trust_both* to QCAM register. See [Table 785, “QCAM - QoS Capabilities Mask Register Layout,”](#) on page 977.
- [Internal] PDDR - Updates units to meet the SFF spec. Renamed *write_offload* to *write_offload_rc* in NVMF capabilities
- Renamed *read_offload* to *read_offload_rc* in NVMF capabilities,
- Renamed *read_write_offload* to *read_write_offload_rc* in NVMF capabilities
- Renamed *read_write_flush_offload* to *read_write_flush_offload_rc* in NVMF capabilities
- Added *write_offload_dc* to NVMF capabilities
- Added *read_offload_dc* to NVMF capabilities
- Added *read_write_offload_dc* to NVMF capabilities,
- Added *read_write_flush_offload_dc* to NVMF capabilities
- Removed *num_port_cap* from NV_GLOBAL_PCI_CAP. See [Table 432, “NV_GLOBAL_PCI_CAP Layout,”](#) on page 686.
- Added *num_ports* to HW Info in MGIR. See [Section 28.26.13, “MPCIR - Management and PHY Control for FW ISSU Flow Register,”](#) on page 1544.
- Added *mtpps_fs* and *mtpps_enhanced_out_periodic_adjustment* to MCAM register. See [Section 28.32.1, “MCAM - Management Capabilities Mask Register,”](#) on page 1643.
- Added *fields_select* and *enhanced_out_periodic_adjustment* to MTPPS register. [Section 22.13.2, “MTPPSE – Management Pulse Per Second Event Register,”](#) on page 1155.
- Added *tunneled_atomic* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *tunneled_atomic* to Mkey Context. See [Table 363, “MKey Context Fields,”](#) on page 631.
- Added *tunneled_atomic* to *mkey_ctx_mask* in UMR control segment. See [Table 53, “UMR Control Segment Fields,”](#) on page 323.
- Added *umr_fence* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *inner_ipv6_hop_limit*, *outer_ipv6_hop_limit*, *inner_ipv4_ttl* and *outer_ipv4_ttl* to Flow Table Fields Supported. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- Added *ttl_hoplimit* to Flow Table Entry Match Set Layer 2-4. See [Table 2125, “Flow Table Entry Match Set Layer 2-4 Field Description,”](#) on page 2006.
- Added OFFLOAD_ERROR event to Affiliated Asynchronous Errors. See [Table 175, “Event Type and Coding,”](#) on page 477.
- Added *icmd_exmb* to ICMD_QUERY_CAPABILITY. See [Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,”](#) on page 2274.

- Added External Mailbox to ICMD Configuration Domain. See Table 358, “ICMD Configuration Domain,” on page 619.
- Added *exmb* to ICMD Control. See Table 359, “ICMD Control Layout,” on page 620.
- Added External Mailbox. See Section 8.25.2.1, “External Mailbox,” on page 677.
- Added *OUT_IPV6_HOPLIMIT* and *IN_IPV6_HOPLIMIT* to field in Set Action. See Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051.
- Updated the following in Section 8.18, “Flow Table,” on page 414:
 - Added new Flow Table actions Encapsulate and Decapsulate and **Modify Header** and defined their execution order.
 - Added description for the Encap header allocation flow.
 - Added description for the **Modify Header Context** allocation flow.
 - Added limitation for Modify Header action for IP header fields in packets where the following headers may be affected.
 - Added clarification that Modify Header action requires no ambiguities in the existence of the header field.
 - Added clarification that **Modify Header** and drop actions are mutually exclusive.
 - Added limitation that Encapsulate (Decapsulate) action in FDB is supported only in Flows going to (from) the Uplink.
 - Added restriction to prevent using the same flow counter in multiple table types.
- Added *fdb_flow_counter_en* to Flow Table E-switch Capabilities. See Table 1586, “Flow Table E-switch Capabilities Fields,” on page 1751.
- Added *inner_esp_spi* and *outer_esp_spi* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Added *inner_esp_spi* and *outer_esp_spi* to Flow Table Entry Match Set Misc Parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Added Quality of Service chapter. See Chapter 19, “Quality of Service” (page 925).
- Added Transmit Scheduler to Software Interface. See Section 8.26, “Transmit Scheduler,” on page 535.
- Added Hairpin section. See Section 4.8, “Hairpin,” on page 215.
- Added *vhca_id*, *hairpin* and *log_max_hairpin_wq_sz* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *hairpin*, *hairpin_peer_vhca* and *hairpin_peer_rq* to SQ Context. See Table 138, “SQ Context Format,” on page 404.
- Added *hairpin*, *hairpin_peer_vhca* and *hairpin_peer_sq* to RQ Context. See Table 124, “RQ Context Format,” on page 385.

- Updated *log_wq_sz* in WQ Context. See Table 127, “Work Queue (WQ) Fields,” on page 391.

Revision 0.43 (February 2017)

- Added *access_other_hca_roce* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *dc_req_sctr_data_cqe* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added QUERY_OTHER_HCA_CAP command. See Section 29.3.6, “QUERY_OTHER_HCA_CAP - Query Other HCA Capabilities,” on page 1807.
- Added MODIFY_OTHER_HCA_CAP command. See Section 29.3.7, “MODIFY_OTHER_HCA_CAP - Modify Other HCA Capabilities,” on page 1809.
- Removed *access_other_function_caps* from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Packet Pacing section to support Burst. See Section •, “ETS TSAR (root) followed by a sequence of DWRR or Round Robin TSARs, so the path (excluding the leaf) is shorter or equal to $2^{\text{HCA\_CAP.log\_esw\_max\_sched\_depth}}$ but holds at least one TSAR,” on page 928.
- Updated description of *host_ready* field. See Table 2749, “ICMD_MH_SYNC Output Mailbox Field Description,” on page 2283.
- Added *packet_pacing_burst_bound* to QoS capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
- Added *burst_upper_bound* to SET_PP_RATE_LIMIT command. See Table 2574, “SET_PP_RATE_LIMIT Input Structure Layout,” on page 2199.
- Added *burst_upper_bound* to QUERY_PP_RATE_LIMIT command. See Table 2582, “QUERY_PP_RATE_LIMIT Output Structure Layout,” on page 2202.
- Added *bth_dst_qp* to Flow Table Entry Match Set Misc Parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Added *bth_dst_qp* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Updated rendezvous description in Tag Matching section. See Section 20.3.1, “Overview,” on page 986.
- Updated Tag Matching CQE Generation. See Section 20.3.3, “Tag Matching CQE Generation,” on page 992.
- Modified protocol field in TMH to operation and *has_tag*. See Table 250, “TMH - Tag Matching Header Layout,” on page 517.
- Added Firmware Components Update. See Chapter 10, “Firmware Components Update” (page 692).
- Added Section 30.8.1, “ICMD_GET_FRC_CHALLENGE - Get Factory Re-customization Token,” on page 2298

- Added Section 30.8.2, “ICMD_SET_FRC - Set Factory Re-customization,” on page 2301
- Fixed *cq_ci* size to 24 bits. See Table 672, “NODNIC CQ Arming Word Layout,” on page 826.
- Updated SW and HW Phase Counters section. See Section 8.21.2.3, “SW and HW Phase Counters,” on page 515.
- Added *message_completion_canceled* to *application_opcode* in CQE format. See Table 147, “Completion Queue Entry Fields,” on page 437.
- Added *cq_period_mode_modify* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *cq_period_mode* to *field_select* of Modify_CQ. See Table 1875, “MODIFY_FIELD_SELECT Structure Field Descriptions,” on page 1896.
- Updated *checksum* field in Mini CQE with checksum to be 16 bits instead of 32 bits. See Table 165, “Responder Mini CQE With Checksum Layout,” on page 463.
- Added *atomic* to ODP per transport Services Capabilities. See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727.
- Added *max_tx_speed* to Query Vport State. See Table 2489, “QUERY_VPORT_STATE Output Structure Layout,” on page 2155.
- Added *dc_v2* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *log_max_guaranteed_connections* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671
- Added *log_max_dct_connections* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *full_handshake* and *cnak_reverse_sl* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *full_handshake* and *cnak_reverse_sl* to CREATE_QP Required Parameters. See Table 1885, “CREATE_QP Required Parameters,” on page 1902.
- Added *reverse_sl* to UD Address Vector. See Table 109, “UD Address Vector Layout,” on page 362.
- Added *force_full_handshake*, *min_guaranteed_connection* and *max_outstanding_connections* to DCT Context. See Table 25, “DCT Context Layout,” on page 279.
- Added *hca_sq_owner* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *multi_path_rc_rdma*, *multi_path_dc_rdma* and *multi_path_xrc_rdma* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Updated *multi_path* description in QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
- Updated WQ offset in XRQ Context to 0x80 instead of 0x140. See Table 246, “XRQ Context Layout,” on page 510.

- Replaced *flow_definition_mode* with *rpg_gd* in Field Select R-Roce RP. See Table 2680, “FIELD_SELECT_R-RoCE RP Structure Field Descriptions,” on page 2245.
- Added Tag Matching CQE Summarize. See Table 792, “Tag Matching CQE Summarize,” on page 993.
- Added *hll_mask* to QHLL register. See Table 263, “QHLL - QoS Head of Queue Lifetime Limit Register Fields,” on page 717.
- Added QP to destination type in Destination Format. See Table 2159, “Destination Entry Format Field Descriptions,” on page 2032.
- Added *disable_local_lb* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *disable_uc_local_lb* and *disable_mc_local_lb* to Nic Vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added *disable_uc_local_lb* and *disable_mc_local_lb* to field_select in Modify Nic Vport Context command, See Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.
- Added *cq_invalidate* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671. .
- Added CQ_INVALID to status in CQ Context. See Table 172, “Completion Queue Context Field Descriptions,” on page 468.
- Added *status* to field_select of Modify_CQ. See Table 1875, “MODIFY_FIELD_SELECT Structure Field Descriptions,” on page 1896.
- Renamed *uar_4kb* to *uar_4k*. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Renamed *number_4k_uars_per_page* to *number_of_uars_per_page* and location was changed in Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *multi_path* to DCT Context. See Table 25, “DCT Context Layout,” on page 279.
- Added *rq_delay_drop* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Drop-Less RQ section. See Section 8.14.3, “Drop-Less RQ,” on page 398.
- Added *delay_drop_en* to RQ context. See Table 124, “RQ Context Format,” on page 385.
- Added *general_notification_event* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added General Notification Event. See Section 8.20.12.8, “General Notification Event,” on page 495.
- Added SET_DELAY_DROP_PARAMS command. See Section 29.15.5, “SET_DELAY_DROP_PARAMS – Set Delay Drop Parameters,” on page 1971.
- Added QUERY_DELAY_DROP_PARAMS command. See Section 29.15.6, “QUERY_DELAY_DROP_PARAMS – Query Delay Drop Parameters,” on page 1972.
- Removed dual port NODNIC. See Chapter 12, “No-DRAM NIC” (page 813).

- Updated [Section 12.3.2, “Port Initialization,”](#) on page 814
- Updated [Section 12.6, “Receiving Packets,”](#) on page 826
- Updated [Section 12.7, “Completions and Interrupts,”](#) on page 826
- Updated [Section 12.5, “Sending Packets,”](#) on page 825
- Updated [Section 12.8, “Enabling/Disabling DMA,”](#) on page 827
- Added *db_record_addr* to Table 664, “NODNIC Ring Configuration Registers Layout,” on page 821.
- Added *support_db_record_rx*, *support_uar_tx_doorbell*, *support_bar_dma_en* and *support_bar_cq_ctrl* fields to Table 660, “NODNIC Configuration Registers Layout,” on page 815.
- Removed InfiniBand-NODNIC. See Chapter 12, “No-DRAM NIC” (page 813).
- Added *send_ring_0_uar_index*, *cq_n*, *cq_dbr_addr*, *dma_en_set* and *dma_en_clr* to Table 662, “NODNIC Port Configuration Registers Layout,” on page 817.
- Added *calc_matrix_type_8bits* to Vector Calc Capabilities. See [Table 1595, “Vector Calc Capabilities Layout,”](#) on page 1772.
- Added *calc_matrix_type* to Vector Calc Segment. See [Table 77, “Vector Calc Segment Format,”](#) on page 336.
- Added *log_max_num_of_staging_buffer_pages* and *log_min_num_of_staging_buffer_pages* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *nvme_offload_en* to QP context. See [Table 19, “QP Context Structure Layout,”](#) on page 258
- Added *nvme_offload_en* to RTR2RTS_QP Optional parameters. See [Table 1911, “RTR2RTS_QP Optional Parameters Mask,”](#) on page 1916.
- Added *nvme_offload_en* to RTS2RTS_QP Optional parameters. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,”](#) on page 1919.
- Added *log_max_rndv_payload_size* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added CQE Format for application offload. See [Section 8.19.1.3, “CQE Format For Application Offload,”](#) on page 448.
- Added *mcam_reg* and *pcam_reg* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671
- Added PCAM to Network Ports Registers. See [Section 28.19.1, “PCAM - Ports Capabilities Mask Register,”](#) on page 1146.
- PDDR - Added Bias fields to module info page and fixed wavelength size issue. [Section 28.19.61, “PDDR - Port Diagnostics Database Register,”](#) on page 1362
- Added *egress_hoq_stall* counter to discard counter group to PPCNT register. See [Section 988, “Ethernet Discard Counters Group Data Layout,”](#) on page 1191.

- Added MCAM to Management and Peripherals Registers. See [Section 28.32.1, “MCAM - Management Capabilities Mask Register,”](#) on page 1643.
- [Internal] Added MQSP for QSA parameters query. [Section 28.26.4, “\[Internal\] MQSP - Module QSa Properties Register,”](#) on page 1534.
- Added MCION for Module’s I/O and Notifications. See [Section 28.26.3, “MCION - Management Cable IO and Notifications Register,”](#) on page 1533.
- Marked MGIR as supported for HCA devices. See [Table 1326, “Management and Peripherals Control Register Summary,”](#) on page 1492.
- Added *manufacturing_base_mac* and *hw_dev_id* to *HW_info* in MGIR register. See [Table 1399, “MGIR - Management General Information Register Layout,”](#) on page 1545.
- Added *rom_type*, *rom_arch* and *rom_version* to *sw_info* in MGIR register. See [Table 1405, “Software Info Layout,”](#) on page 1550.
- Updated explanations of MTPPS and added support of virtual pin. See [Section 28.30.1, “MTPPS – Management Pulse Per Second Register,”](#) on page 1611.
- [Internal] Added MTUTC for future devices. See [Section 28.30.3, “MTUTC – Management UTC Register,”](#) on page 1617.
- Removed shadow from Configuration Item. See [Table 1444, “Configuration Item Layout,”](#) on page 1572.
- Moved *writer_id* and *writer_host_id* to offset 0x0 in Configuration Item. See [Table 1444, “Configuration Item Layout,”](#) on page 1572.
- Added *panic_teartdown* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added `ALLOC_MODIFY_HEADER_CONTEXT` command. See [Section 29.18.18, “ALLOC_MODIFY_HEADER_CONTEXT - Allocate Modify Header Context,”](#) on page 2048.
- Added `DEALLOC_MODIFY_HEADER_CONTEXT` command. See [Section 29.18.19, “DEALLOC_MODIFY_HEADER_CONTEXT - De-Allocate Modify Header Context,”](#) on page 2054.
- Added `QUERY_MODIFY_HEADER_CONTEXT` command. See [Section 29.18.20, “QUERY_MODIFY_HEADER_CONTEXT – Query Modify Header Context,”](#) on page 2055.
- Added `MODIFY_HEADER` to action in Flow Context. See [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.
- Added *modify_header_id* to Flow Context. See [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.
- Added *log_max_modify_header_context* to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Added *max_modify_header_actions* to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Added *header_modify_nic_receive* and *header_modify_nic_transmit* to Flow Table NIC Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.

- Added *header_modify_esw_fdb*, *header_modify_esw_acl_ingress* and *header_modify_esw_acl_egress* to Flow Table E-switch Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748.
- Added *outer_ip_ttl* and *inner_ip_ttl* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Removed *kdnet_ctrl* from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Removed KDNET_CTRL command.
- Added *multi_prio_sq* to nic vport context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added *multi_prio_sq* to field select in MODIFY_NIC_VPOR_CONTEXT command. See Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.
- Added *multi_prio_sq* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *max_geneve_opt_len* to Per Protocol Networking Offload. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *outer_geneve_opt_len* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Added *geneve_opt_len* to Flow Table Entry Match Set Misc Parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Added *rnr_nak_q_counters* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *resp_rnr_nak* and *req_rnr_retries_exceeded* to Q Counters. See Section 29.35.3, “QUERY_Q_COUNTER – Query Q Counter,” on page 2188.
- Added *Reset_Sandbox* to operation in FPGA_CTRL register. See Table 896, “FPGA_CTRL Register Fields,” on page 1110.
- Renamed *vlan_cap* to *vlan_strip* in Per Protocol Network Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *wqe_vlan_insert* to Per Protocol Network Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *insert_vlan* to Eth Segment. See Table 48, “Eth Segment Fields,” on page 319
- Updated VLAN Insertion/Stripping section. See Section 4.3.10, “VLAN Insertion/Stripping,” on page 201.
- Added *nsid* and *nsid_valid* to NVMe target offload context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Renamed *log_max_num_of_staging_buffer_pages* to *log_max_staging_buffer_pages* in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Renamed *log_min_num_of_staging_buffer_pages* to *log_min_staging_buffer_pages* in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *outer_ip_version* and *inner_ip_version* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Added *ip_version* to Flow Table Entry Match Set Layer 2-4. See Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005.
- Removed *clear_int* from Initialization Segment. See Section 5.4, “Initialization Segment,” on page 234.
- Added *enhanced_multi_pkt_send_wqe* to Per Protocol Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added Enhanced Multi Packet Send WQE section. See Section 4.3.5, “Enhanced Multi-Packet Send WQE Offload,” on page 193.
- Added ENHANCED_MPSW to opcode in Ctrl Segment. See Table 43, “General Ctrl Segment Field,” on page 313.
- Added NV_POWER_CONF. See Section 11.7.94, “NV_POWER_CONF,” on page 763.
- Added NV_POWER_CAP. See Section 11.7.95, “NV_POWER_CAP,” on page 765.
- Added *num_port_cap* to NV_GLOBAL_PCI_CAP. See Section 11.7.29, “NV_GLOBAL_PCI_CAP,” on page 686.
- Increased *max_num_pfs* to 8bit in NV_GLOBAL_PCI_CAP. See Section 11.7.29, “NV_GLOBAL_PCI_CAP,” on page 686.
- Increased *num_pfs* to 8bit in NV_GLOBAL_PCI_CONF. See Section 11.7.28, “NV_GLOBAL_PCI_CONF,” on page 682.
- Added *one_qp_per_recovery* in NV_SW_OFFLOAD_CONF. See Section 11.7.35, “NV_SW_OFFLOAD_CONFIG,” on page 702.

Revision 0.42 (September 2016)

- Added *gid_table_size_ro* and *pkey_table_size_ro* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *modify_rq_counters_set_id* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Updated InfiniBand Network Virtualization to support LID-based. See Section 14.6, “InfiniBand Network Virtualization,” on page 849.
- Added *grh_required* to field_select in HCA_VPORT context. See Table 37, “HCA_Vport Context Field Descriptions,” on page 296.
- Added CAPI section. See Section 22, “CAPI- Coherent Accelerator Programming Interface,” on page 1035.
- Added *log_pasid_size* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Added *pasid* to ALLOC_UAR command. See Table 1706, “ALLOC_UAR Input Structure Layout,” on page 1834.
- Added *pasid* to SET_DC_CNAK_TRACE command. See Table 2478, “SET_DC_CNAK_TRACE Input Structure Layout,” on page 2151.
- Added *pasid* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *pasid* to XRC_SRQ context. See Table 114, “XRC_SRQ Context Layout,” on page 370.
- Added *pasid* to WQ context. See Table 126, “Work Queue (WQ) Format,” on page 389.
- Added *pasid* to CQ context. See Table 171, “Completion Queue Context Layout,” on page 467.
- Added *pasid* to EQ context. See Table 244, “Event Queue Context Layout,” on page 505.
- Added *pasid* to Mkey context. See Table 362, “MKey Context Format,” on page 629.
- Added *manage_pages_pasid* and *cmd_pages* to Initialization segment. See Table 14, “Initialization Segment,” on page 234.
- Added Sandbox_Bypass_On and Sandbox_Bypass_Off to operation in FPGA_CTRL. See Table 896, “FPGA_CTRL Register Fields,” on page 1110.
- Added *watchdog_failure* to FPGA_ERROR Event. See Table 240, “Event_data Field - FPGA Error Event Layout,” on page 504.
- Added *tunnel_stateless_geneve_rx* to Per Protocol Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *outer_geneve_vni* and *outer_geneve_oam* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Added *geneve_vni* and *geneve_oam* to Flow Table Entry Misc parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Added Remote Micro-Applications section. See Chapter 25, “Remote Micro-Applications” (page 1042).
- Added *micro_atomic_operations* to atomic_operations field in Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728.
- Added *load_remote_micro_app_4byte*, *load_remote_micro_app_8byte*, *store_remote_micro_app_4byte* and *store_remote_micro_app_8byte* to Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728.
- Updated *atomic_mode* to support remote micro applications. See Table 20, “QP Context Field Descriptions,” on page 261.
- Added LOAD_REMOTE_MICRO_APP and STORE_REMOTE_MICRO_APP to Opcode field in General Ctrl segment. See Table 43, “General Ctrl Segment Field,” on page 313.

- Added *remote_micro_app_operation_code* to Ctrl segment. See Table 43, “General Ctrl Segment Field,” on page 313.
- Added *LOAD_REMOTE_MICRO_APP* and *STORE_REMOTE_MICRO_APP* to Wqe construction summary. See Table 106, “WQE Construction Summary,” on page 355.
- Added Global Atomicity Guaranteeing section. See Chapter 26, “Global Atomicity Guaranteeing (GAG)” (page 1046).
- Added *uar_4kb* and *number_4k_uars_per_page* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *umr_extended_translation_offset* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *translation_offset_42_16* to UMR Control Segment. See Table 52, “UMR Control Segment Layout,” on page 323.
- Renamed *translation_offset* to *translation_offset_15_0* to UMR Control Segment. See Table 52, “UMR Control Segment Layout,” on page 323.
- Added *null_mkey* to QUERY_SPECIAL_CONTEXT command. See Section 29.6.5, “QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers,” on page 1871.
- Added *allow_sniffer_and_nic_rx_shared_tir* to Flow Table Nic Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
- Added Quality Of Service section. See Section 2.6, “Quality of Service,” on page 179.
- Added *esw_scheduling*, *esw_tsar_type*, *max_qos_para_vport*, *esw_element_type* and *max_tsar_bw_share* to QoS capabilities. See Section 29.3.4.9, “Quality of Service Capabilities,” on page 1773.
- Added Transmit Scheduler to Software Interface. See Section 8.26, “Transmit Scheduler,” on page 535.
- Added *qos_para_vport_number* to SQ context. See Table 138, “SQ Context Format,” on page 404.
- Added *qos_para_vport_number* to CREATE_SQ and MODIFY_SQ Bitmask. See Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407.
- Added *CREATE_SCHEDULING_ELEMENT*, *DESTROY_SCHEDULING_ELEMENT*, *QUERY_SCHEDULING_ELEMENT*, *MODIFY_SCHEDULING_ELEMENT*, *CREATE_QOS_PARA_VPORT*, *CREATE_QOS_PARA_VPORT* commands, See Section 29.36, “QoS Commands - Quality of Service Commands,” on page 2199.
- Added *outer_geneve_protocol_type* to Flow Table Fields Supported. See Table 1591, “Flow Table Fields Supported Format,” on page 1761.
- Added *geneve_protocol_type* to Flow Table Entry Match Set Misc Parameters. See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.

- Removed *qpn*, *rdma*, *read_write* and *requester* fields from WQE Associated Page Fault Event. See Table 189, “Event_data Field - WQE Associated Page Fault Event Layout,” on page 484.
- Added *token*, *wq_number* and *page_fault_type* to WQE Associated Page Fault Event. See Table 189, “Event_data Field - WQE Associated Page Fault Event Layout,” on page 484.
- Removed *qpn*, *rdma*, *read_write* and *requester* fields from RDMA Page Fault Event. See Table 191, “Event_data Field - RDMA Page Fault Event Layout,” on page 485.
- Added *token* and *page_fault_type* to RDMA Associated Page Fault Event. See Table 191, “Event_data Field - RDMA Page Fault Event Layout,” on page 485.
- Removed *qpn*, *rdma*, *read_write* and *requester* fields from Page Fault Resume command. See Table 1815, “PAGE_FAULT_RESUME Input Structure Layout,” on page 1869.
- Added *token*, *wq_number* and *page_fault_type* to PAGE_FAULT_RESUME command. See Table 1815, “PAGE_FAULT_RESUME Input Structure Layout,” on page 1869.
- Modified fields in ODP per transport service to be RW. See Table 1576, “ODP per Transport Service Capabilities Field Descriptions,” on page 1728
- Added *striding_rq* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Striding RQ section. See Section 8.9.1.3, “Striding Receive WQ,” on page 304.
- Added LINKED_LIST_STRIDING_RQ to *wq_type* in WQ. See Table 127, “Work Queue (WQ) Fields,” on page 391.
- Added *single_wqe_log_num_of_strides*, *two_byte_shift_en* and *single_stride_log_num_of_bytes* to WQ. See Table 127, “Work Queue (WQ) Fields,” on page 391.
- Updated *byte_cnt* in CQE format to support Striding RQ. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435.
- Updated *wqe_counter* in CQE format to support Striding RQ. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435.
- Added *wqe_id* to CQE format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435.
- Renamed SET_DIAGNOSTIC command to SET_DIAGNOSTIC_PARAMS, See Section 29.40.1, “SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters,” on page 2222.
- Added QUERY_DIAGNOSTIC_PARAMS command. See Section 29.40.2, “QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters,” on page 2224.
- Renamed QUERY_DIAGNOSTIC command to QUERY_DIAGNOSTIC_COUNTERS. See Section 29.40.3, “QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters,” on page 2226.
- Changed image to flash in FPGA_CTRL register. See Table 895, “FPGA_CTRL Register Layout,” on page 1108.

- Added Auxiliary Pulse-Per-Second (PPS) Signals section. See Section 13.3.1.1, “Auxiliary Pulse-Per-Second (PPS) Signals,” on page 833.
- Added *pps* and *pps_modify* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added PPS Event. See Section 8.20.12.6, “PPS Event - Pulse Per Second Event,” on page 494.
- Added MTPPS and MTPPSE registers. See Section 28.30, “Pulse Per Second Registers,” on page 1611.
- Added *image_changed* to FPGA Error Event. See Table 240, “Event_data Field - FPGA Error Event Layout,” on page 504.
- Resized *staging_buffer_page_offset*, *nvme_cq_page_offset* and *nvme_cq_page_offset* in NVMe-oF Target Offload Context Header. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Renamed *memory_page_size* to *memory_log_page_size* in NVMe-oF Target Offload Context Header. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Updated description and rename *iocapsz* to *ioccsz* in NVMe-oF Target Offload Context Header. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Updated description of *icdoff* in NVMe-oF Target Offload Context Header. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Added *log_min_ioccsz* and *max_icdoff* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Moved WQ to offset 0x140 in XRQ context. See Table 246, “XRQ Context Layout,” on page 510.
- Added *esw_bw_share* and *esw_rate_limit* to QoS capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
- Added Head of Queue Lifetime section. See Section 17.3, “Head of Queue Lifetime,” on page 820.
- Added *hll* to QoS capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
- Added QHLL register. See Section 21.4.4, “QHLL - QoS Head of Queue Lifetime Limit Register,” on page 717.
- Added CQE Compressing section. See Section 8.19.9, “CQE Compressing,” on page 459.
- Added *fw_ctrl_update* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *fw_ctrl_update_icmd* to ICMD_QUERY_CAPABILITY. See Table 2737, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Field Description,” on page 2275.

Revision 0.41 (June 2016)

- Renamed *diagnostic_counters* to debug in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *num_of_diagnostic_counters* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Renamed Diagnostic Counters Capabilities to Debug Capabilities. See Section 29.3.4.10, “Debug Capabilities,” on page 1777.
- Added *health_mon_rx_activity* to Debug Capabilities. See Section 29.3.4.10, “Debug Capabilities,” on page 1777.
- Added RX Path Activity section. See Section 26.5.1, “RX Path Activity,” on page 988.
- Added *on_demand* and *enable* to SET_DIAGNOSTICS command. See Table 2640, “SET_DIAGNOSTIC_PARAMS Input Structure Layout,” on page 2222.
- Updated counters offset from 10h to 20h. See Table 2640, “SET_DIAGNOSTIC_PARAMS Input Structure Layout,” on page 2222.
- Renamed *log_num_of_samples* to *num_of_samples* in QUERY_DIAGNOSTICS. See Table 2650, “QUERY_DIAGNOSTIC_COUNTERS Input Structure Layout,” on page 2226.
- Increased size of *num_of_samples* in QUERY_DIAGNOSTICS. See Table 2650, “QUERY_DIAGNOSTIC_COUNTERS Input Structure Layout,” on page 2226.
- Replaced *sample* with *diag_counter* in QUERY_DIAGNOSTICS. See Table 2652, “QUERY_DIAGNOSTIC_COUNTERS Output Structure Layout,” on page 2226.
- Replaced *time_stamp* with *sample_id* in Diagnostic counter. See Table 2654, “Diagnostic Counter Structure Layout,” on page 2227.
- Added Software Parsing Offload section. See Section 4.3.3, “Software Parsing - SWP,” on page 191.
- Added *swp_** fields to Eth Segment. See Table 47, “Eth Segment Format,” on page 318.
- Added *allow_swp* to SQ context. See Table 138, “SQ Context Format,” on page 404.
- Added *swp*, *swp_csum* and *swp_iso* to Per Protocol Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *mlnx_tag_ethertype* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added MLNX ETHERTYPE TAGGING section. See Section 4.7, “MLNX ETHERTYPE TAGGING,” on page 214.
- Added FPGA Device chapter. See Chapter 21, “FPGA Device” (page 1011).
- Added FPGA Events. See Section 8.20.15, “FPGA Events,” on page 504.
- Added *max_fpga_qp_msg_size* to Shell Caps. See Table 893, “Shell_Caps Layout,” on page 1107.

- Added `FPGA_QUERY_QP_COUNTERS` command. See Section 29.20.5, “FPGA_QUERY_QP_COUNTERS - Query FPGA QP Counters,” on page 2068.
- Added `FPGA_SHELL_COUNTERS` register. See Section 28.8.4, “FPGA_SHELL_COUNTERS Register,” on page 1112.
- Added *fpga* to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *fpga* to `ICMD_QUERY_CAPABILITIES`. See Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274.
- Added FPGA registers. See Section 28.8, “FPGA Registers,” on page 1103.
- Added FPGA QP commands. See Section 29.20, “FPGA QP Commands,” on page 2063.
- Added FPGA QP Context. See Section 8.25, “FPGA QP,” on page 530.
- Modified size of NVMe context in XRQ. See Table 246, “XRQ Context Layout,” on page 510.
- Added *sqt_dbr_addr* and *cqh_dbr_addr* to NVMe context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Updated fields location in NVMe context. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.
- Updated the pages order in NVMe section. See Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999.
- Added *lwm* to `CREATE_RQ` and `MODIFY_RQ` bitmask. See Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395.
- Added *extended_retry_count* to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *extended_retry_count_valid*, *extended_retry_count* and *extended_cur_retry_count* to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *extended_retry_count_valid* and *extended_retry_count* to `RTR2RTS_QP` Required parameters. See Table 1910, “RTR2RTS_QP Required Parameters,” on page 1916.
- Added *dcbx* to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *scatter_fcs* to Per protocol Networking Offload capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *scatter_fcs* to RQ Context. See Table 124, “RQ Context Format,” on page 385.
- Added *scatter_fcs* to `MODIFY_RQ` bitmask. See Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395.
- Added tag matching and rendezvous offload section. See Section 20.3, “Tag Matching and Rendezvous Offload,” on page 986.
- Added *tag_matching*, *rndv_offload_rc*, *rndv_offload_dc*, *log_max_xrq* and *log_tag_matching_list_sz* to `HCA_CAP`. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added XRQ Matching Context. See Section 8.21, “XRQ Context,” on page 509.

- Added Tag Matching segment summary to Send WQE format. See Section 8.9.4.1, “Send WQE Format,” on page 310.
- Added *Tag_Matching* to opcode in Ctrl Segment. See Table 43, “General Ctrl Segment Field,” on page 313.
- Added Tag Matching Segment. See Section 8.9.4.1.17, “Tag Matching Segment,” on page 342.
- Added Tag Matching to WQE Construction Summary. See Table 106, “WQE Construction Summary,” on page 355.
- Added *xrqn* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *xrqn* to *rq_type* in QP context. See Table 20, “QP Context Field Descriptions,” on page 261.
- Added *xrqn* to DCT context. See Table 25, “DCT Context Layout,” on page 279.
- Added *application*, *application_opcode* and *application_info* to CQE format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435
- Added XRQ Commands. See Section 29.27, “XRQ Commands,” on page 2116.
- Added NVMe Target offload Context. See Section 8.21.3, “NVM Express over Fabric (NVMe-oF) Target Offload Context,” on page 519.
- Added *nvme_target_offload*, *nvme_write_offload*, *nvme_read_offload*, *nvme_read_write_offload*, *nvme_read_write_flush_offload* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added NVMe over fabric target application offload section. See Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999.
- Added XRQ to ARM_RQ command. See Section 29.25.4, “ARM_RQ,” on page 2108
- Removed ARM_XRQ command. See Section 29.27, “XRQ Commands,” on page 2116
- Removed *rndv* and *tag_matching* fields from QP Context. See Table 19, “QP Context Structure Layout,” on page 258
- Removed *rndv* and *tag_matching* fields from DCT Context. See Table 25, “DCT Context Layout,” on page 279.
- Added No packet operation to opcode in cqe format. See Table 147, “Completion Queue Entry Fields,” on page 437.
- Removed *cqtdb* and *sqhdb* from required PAS list for NVME. See Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999.

Revision 0.40 (April 2016)

- Added *umr_ptr_rlky* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Added *reg_umr_sq* to Per Protocol Networking Offload Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Added *reg_umr* to SQ context. See [Table 138, “SQ Context Format,”](#) on page 404.
- Added *disable_link_up_by_init_hca* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *imaicl* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Updated User-Mode Memory registration. See [Section 18.3.1, “User-Mode Memory Registration \(UMR\),”](#) on page 918.
- Removed Dropped Packet Logging.
- Removed Get Dropped Packet Log command.
- Added *max_encap_header_size* to E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- Added *log_max_encap_headers* to E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- Added *vlan_encap_decap* and *nvgre_encap_decap* to E-switch Capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- Added *encap* and *decap* to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Updated [Section 8.18.4.7, “E-Switch Forwarding Data Base \(FDB\),”](#) on page 430.
- Added `ALLOC_ENCAP_HEADER` command. See [Section 29.18.15, “ALLOC_PACKET_REFORMAT_CONTEXT - Allocate Packet Reformat Context,”](#) on page 2043.
- Added `DEALLOC_ENCAP_HEADER` command. See [Section 29.18.16, “DEALLOC_PACKET_REFORMAT_CONTEXT - De-Allocate Packet Reformat Context,”](#) on page 2046.
- Added `QUERY_ENCAP_HEADER` command. See [Section 29.18.17, “QUERY_PACKET_REFORMAT_CONTEXT – Query Packet Reformat Context,”](#) on page 2047.
- Added ENCAP and DECAP actions to Flow Context. See [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.
- Added *encap_id* to Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,”](#) on page 2025.
- Added Flow Table Context. See [Table 2102, “FLOW TABLE CONTEXT Structure Layout,”](#) on page 1989.
- Added *encap_ptr* to *modify_enable_mask* in `SET_FLOW_TABLE_ENTRY` command. See [Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,”](#) on page 2023.
- Added tag matching and rendezvous offload section. See [Section 20.3, “Tag Matching and Rendezvous Offload,”](#) on page 986.

- Added *tag_matching*, *rndv_offload_rc*, *rndv_offload_dc*, *log_max_xrq* and *log_tag_matching_list_sz* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added XRQ Matching Context. See Section 8.21, “XRQ Context,” on page 509.
- Added Tag Matching segment summary to Send WQE format. See Section 8.9.4.1, “Send WQE Format,” on page 310.
- Added Tag_Matching to opcode in Ctrl Segment. See Table 43, “General Ctrl Segment Field,” on page 313.
- Added Tag Matching Segment. See Section 8.9.4.1.17, “Tag Matching Segment,” on page 342.
- Added Tag Matching to WQE Construction Summary. See Table 106, “WQE Construction Summary,” on page 355.
- Added *rndv*, *tag_matching* and *xrqn* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *xrqn* to *rq_type* in QP context. See Table 20, “QP Context Field Descriptions,” on page 261.
- Added *rndv*, *tag_matching* and *xrqn* to DCT context. See Table 25, “DCT Context Layout,” on page 279.
- Added *application*, *application_opcode* and *application_info* to CQE format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435
- Added XRQ Commands. See Section 29.27, “XRQ Commands,” on page 2116.
- Added NVMf Target offload Context. See Section 8.21.3, “NVM Express over Fabric (NVMe-oF) Target Offload Context,” on page 519.
- Added *nvme_target_offload*, *nvme_write_offload*, *nvme_read_offload*, *nvme_read_write_offload*, *nvme_read_write_flush_offload* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added NVMe over fabric target application offload section. See Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999.
- Added POST_SEND_OFFLOADED_QP command. See Section 24.23, “DCT Commands,” on page 1183.
- Updated Section 8.18.4.7, “E-Switch Forwarding Data Base (FDB),” on page 430
- Updated Section 8.18.4.8, “E-Switch Ingress Access Control List (ACL),” on page 431
- Added per priority discard counters. See Section 28.19.12, “PPCNT - Ports Performance Counters Register,” on page 1171.
- [Internal] Added FW implementation notes to the PPCNT
- Added modify_tis field to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- [Internal] New register SLRD
- Moved AN disabled capability (`an_disable_cap`) from bit 31 to bit 29. See [Section 28.19.5, “PTYS - Port Type and Speed Register,”](#) on page 1152.
- [Internal] PCMR - Change bit-mask structure
- Removed `dc_hs` from QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258
- Added `temp_warn_event` to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added `Temp_Warning_Event`. See [Section 8.20.12.7, “Temp Warning Event,”](#) on page 495.
- Added Enhanced ULP Offloads section. See [Section 4.6.1.2, “Enhanced ULP Offloads,”](#) on page 210
- Updated The Root Flow Table section. See [Section 8.18.3.2, “Selecting the Root Flow Table,”](#) on page 420
- Added reset to default option to opcode of SET_ROOT_FLOW_TABLE command, See [Table 2113, “SET_FLOW_TABLE_ROOT - Input Structure Field Descriptions,”](#) on page 1996
- Added `underlay_qpn` to SET_ROOT_FLOW_TABLE command, See [Table 2112, “SET_FLOW_TABLE_ROOT Input Structure Layout,”](#) on page 1995
- Added Enhanced IpoIB to *`ulp_stateless_offload_mode`* in QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added `sl` and `underlay` to TIS Context. See [Table 121, “TIS Context Format,”](#) on page 382.
- Added *`ipoib_enhanced_offloads`* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added IpoIB Enhanced Offload Capabilities to opcode of QUERY_HCA_CAP, See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667
- Added IpoIB Enhanced Offload Capabilities to opcode of SET_HCA_CAP, See [Table 1628, “SET_HCA_CAP Input Structure Field Descriptions,”](#) on page 1804
- Updated `flow_tag` in Flow Context. See [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.
- Added DCBX Change Event. See [Section 8.20.12.5, “DCBX Change Event,”](#) on page 493.
- Added *`operation_type`*, *`cap_local_admin`* and *`cap_remote_admin`* to QETCR. See [Table 731, “QETCR - QoS ETS TC Configuration Register Layout,”](#) on page 949.
- Added *`operation_type`*, *`cap_local_admin`* and *`cap_remote_admin`* to QTCT. See [Section 21.3.3, “QTCT - QoS Switch Traffic Class Table,”](#) on page 714.
- Added DCBX_PARAM Register. See [Section 28.3.4, “DCBX_PARAM Register,”](#) on page 1086.
- Added DCBX_APPLICATION Register. See [Section 28.3.6, “DCBX_APPLICATION Register,”](#) on page 1090.

- Added *log_max_flow_counter_bulk* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *num_of_counters* to Query Flow Counter command. See [Table 2182, “QUERY_FLOW_COUNTER Input Structure Layout,”](#) on page 2041.
- Added list of flow statistics to Query Flow Counter command. See [Table 2184, “QUERY_FLOW_COUNTER Output Structure Layout,”](#) on page 2042.
- Added *access_other_functions_caps* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *function_id* to QUERY_HCA_CAP. See [Table 1567, “QUERY_HCA_CAP Input Structure Layout,”](#) on page 1667.
- Added *function_id* to SET_HCA_CAP. See [Table 1627, “SET_HCA_CAP Input Structure Layout,”](#) on page 1803.
- Added *led_type* and *cap_local_or_uid_only* to MLCR. See [Table 1433, “MLCR - Management LED Control Register Layout,”](#) on page 1566.
- Added *writer_id* and *writer_host_id* to configuration item header. See [Table 1445, “Configuration Item Field Descriptions,”](#) on page 1572.
- Replaced Type with *type_class* and *type_index* in configuration item header. See [Table 1445, “Configuration Item Field Descriptions,”](#) on page 1572.
- Added log to *type_class* in configuration item header. See [Table 1445, “Configuration Item Field Descriptions,”](#) on page 1572.
- Added MNVGN register. See [Table 1469, “MNVGN Register Layout,”](#) on page 1581
- Added MGNLE register. See [Table 1471, “MGNLE Register Layout,”](#) on page 1583
- Added MRTC register. See [Table 1423, “MRTC - Management Real Time Clock Register Layout,”](#) on page 1561.
- Added *fixed_buffer_size* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added Fixed Buffer Size to *access_mode* in Mkey Context. See [Table 363, “MKey Context Fields,”](#) on page 631.
- Renamed *log_page_size* to *log_entity_size* in Mkey Context. See [Table 363, “MKey Context Fields,”](#) on page 631.
- Updated UMR Memory Buffer to supported Fixed Buffer Size. See [Table 57, “UMR Memory Buffer \(KLM\) Fields Description,”](#) on page 326.
- Added *pulse_gen_control* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Renamed *multi_path_tirs* to *nic_rx_multi_path_tirs*. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.

- Added *nic_rx_multi_path_tirs_fts* to Flow Table Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.
- Updated Flow Table Level section to support *nic_rx_multi_path_tirs_fts*. See [Section 8.18.3.3, “Flow Table Level,”](#) on page 421.
- Added LAG Section. See [Section 14, “LAG - Link Aggregation,”](#) on page 399.
- Added LAG commands. See [Section 29.4, “LAG Commands,”](#) on page 1849.
- Added *num_lag_ports* and *lag_master* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *strict_lag_tx_port_affinity* and *lag_tx_port_affinity* to TIS context. See [Table 121, “TIS Context Format,”](#) on page 382.
- Added *lag_tx_port_affinity* to QP context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added *lag_tx_port_affinity* to RST2INIT_QP Required Parameters. See [Table 1894, “RST2INIT_QP Required Parameters,”](#) on page 1906.
- Removed *CFI/DEI* from QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Removed *CFI/DEI* from DCT Context. See [Table 25, “DCT Context Layout,”](#) on page 279.
- Added *lag_demux_flow_table* to op_mod of CREATE_FLOW_TABLE. See [Table 2099, “CREATE_FLOW_TABLE - Input Structure Field Description,”](#) on page 1987.
- Added *lag_master_next_table_id* to Flow Table Context. See [Table 2102, “FLOW TABLE CONTEXT Structure Layout,”](#) on page 1989.
- Added *lag_master_next_table_id* to *modify_field_select* of MODIFY_FLOW_TABLE. See [Table 2105, “MODIFY_FLOW_TABLE - Input Structure Field Description,”](#) on page 1992.
- Added ICMD_MH_SYNC. See [Section 30.4.3, “ICMD_MH_SYNC - Multi-Host Synchronization,”](#) on page 2280.
- Added ICMD_MH_SYNC_STATUS. See [Section 30.4.4, “ICMD_MH_SYNC_STATUS - Multi-Host Synchronization Status,”](#) on page 2283.
- Added ICMD_KDNET_CTRL.
- Added *mh_sync* and *kdnet_ctrl* to ICMD_QUERY_CAPABILITY. See [Section 30.4.1, “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities,”](#) on page 2273.
- Added *port_checks* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *flow_table_properties_nic_receive_rdma* and *flow_table_properties_nic_transmit_rdma* to Flow Table Nic Capabilities. See [Table 1583, “Flow Table NIC Capabilities Layout,”](#) on page 1742.
- Added NIC_RX_RDMA and NIC_TX_RDMA to table type in Flow Table commands. See [Section 29.18, “Flow Table Commands,”](#) on page 1985.

- Updated [Section 27.31.1, “Packet Pacing Commands,”](#) on page 1078.
- Removed *packet_pacing_rate_limit_index* from QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Removed *packet_pacing_rate_limit_index* from INIT2RTR required parameters. See [Table 1904, “INIT2RTR_QP Required Parameters,”](#) on page 1911.
- Removed *packet_pacing_rate_limit_index* from INIT2RTR required parameters. See [Table 1904, “INIT2RTR_QP Required Parameters,”](#) on page 1911.
- Removed *packet_pacing_rate_limit_index* from RTR2RTS optional parameters. See [Table 1911, “RTR2RTS_QP Optional Parameters Mask,”](#) on page 1916.
- Removed *packet_pacing_rate_limit_index* from RTS2RTS optional parameters. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,”](#) on page 1919.
- Added Diagnostic Counters section, See TBD - can’t find it in book.
- Added *diagnostic_counters* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671
- Added Diagnostic Counters Capabilities. See [Section 29.3.4.10, “Debug Capabilities,”](#) on page 1777.
- Added Diagnostic Counters commands. See [Section 29.40, “Diagnostic Counters Commands,”](#) on page 2221.

Revision 0.30 (December 2015)

- Added ULP Stateless Offloads section. See [Section 4.6, “ULP Stateless Offloads,”](#) on page 209.
- Added *ipoib_basic_offloads* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added IPoIB Basic Offload Capabilities to *op_mod* in QUERY_HCA_CAP. See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667.
- Added IPoIB Basic Offload Capabilities to *op_mod* in SET_HCA_CAP. See [Table 1628, “SET_HCA_CAP Input Structure Field Descriptions,”](#) on page 1804.
- Added *ulp_stateless_offload_mode* to QP context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added *ulp_stateless_offload_mode* to CREATE_QP Required parameters. See [Table 1885, “CREATE_QP Required Parameters,”](#) on page 1902.
- Added Vector Calc segment. See [Section 8.9.4.1.13, “Vector Calc Segment,”](#) on page 336.
- Added *vector_calc* field to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.

- Added Vector-Calc Operations section. See [Section 20.2, “Vector-Calc Operations,”](#) on page 986.
- Added Vector Calc capabilities. See [Section 29.3.4.8, “Vector Calc Capabilities,”](#) on page 1772.
- Added *flow_table_modify* to Flow Table Properties. See [Table 1588, “Flow Table Properties Field Descriptions,”](#) on page 1754.
- Added *identified_miss_table* to Flow Table Properties. See [Table 1588, “Flow Table Properties Field Descriptions,”](#) on page 1754.
- Added *table_miss_action* and *table_miss_id* to CREATE_FLOW_TABLE command. See [Section 29.18.1, “CREATE_FLOW_TABLE - Allocate a New Flow Table,”](#) on page 1986.
- Added *table_miss_action* and *table_miss_id* to QUERY_FLOW_TABLE command. See [Section 29.18.5, “QUERY_FLOW_TABLE - Query Flow Table,”](#) on page 1998.
- Added MODIFY_FLOW_TABLE command. See [Section 29.18.2, “MODIFY_FLOW_TABLE - Modify a Flow Table,”](#) on page 1991.
- Added *qos* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added Quality of Service capabilities. See [Table 29.3.4.9, “Quality of Service Capabilities,”](#) on page 1773.
- Added Packet Pacing commands. See [Section 27.31.1, “Packet Pacing Commands,”](#) on page 1078.
- Added *packet_pacing_rate_limit_index* to SQ Context. See [Table 138, “SQ Context Format,”](#) on page 404.
- Added *packet_pacing_rate_limit_index* to CREATE_SQ and MODIFY_SQ Bitmask. See [Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,”](#) on page 407.
- Added *packet_pacing_rate_limit_index* to QP Context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added *packet_pacing_rate_limit_index* to INIT2RTR required parameters. See [Table 1904, “INIT2RTR_QP Required Parameters,”](#) on page 1911.
- Added *packet_pacing_rate_limit_index* to INIT2RTR required parameters. See [Table 1904, “INIT2RTR_QP Required Parameters,”](#) on page 1911.
- Added *packet_pacing_rate_limit_index* to RTR2RTS optional parameters. See [Table 1911, “RTR2RTS_QP Optional Parameters Mask,”](#) on page 1916.
- Added *packet_pacing_rate_limit_index* to RTS2RTS optional parameters. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,”](#) on page 1919.
- Added Expansion ROM section. See [Section 5.3.1.4, “Expansion ROM,”](#) on page 230.
- Added ICMD and Expansion ROM to space in Vendor Specific Capability. See [Table 10, “Vendor Specific Capability Field Descriptions,”](#) on page 228.

- Added MFPA, MFBA and MFBE registers. See [Section 28.25, “Flash Registers,”](#) on page 1520.
- Renamed *eth_prio_primary_in_rts2rts* to *rts2rts_primary_eth_prio* in Roce capabilities. See [Table 1579, “RoCE Capabilities Layout,”](#) on page 1732.
- Added *rts2rts_src_addr_index_for_vlan_valid_vlan_id* to Roce capabilities. See [Table 1579, “RoCE Capabilities Layout,”](#) on page 1732.
- Added option to modify *sl* and *src_addr_index* to RTS2RTS optional parameters. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,”](#) on page 1919.
- Added *rts2rts_primary_sl* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *nic_vport_node_guid_modify* and *nic_vport_port_guid_modify* to E-switch capabilities. See [Table 1593, “E-switch Capabilities Layout,”](#) on page 1768.
- Added option to modify *port_guid* and *node_guid* to MODIFY_NIC_VPORT_CONTEXT. See [Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,”](#) on page 2164.
- Added Cable is Shorted error type to Port Module Event. See [Table 207, “Event_data Field - Port Module Event Layout,”](#) on page 492.
- Added *allow_icmd_access_reg_on_all_registers* to ICMD_ACCESS_CAPABILITY command. See [Section 30.4.1, “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities,”](#) on page 2273.
- Removed *atomic_req_endianness* from Atomic capabilities. See [Table 1577, “Atomic Capabilities Layout,”](#) on page 1728.
- Added *atomic_req_8b_endianness_mode* to Atomic capabilities. See [Table 1577, “Atomic Capabilities Layout,”](#) on page 1728.
- Added *supported_atomic_req_8B_endianness_mode_1* to Atomic capabilities. See [Table 1577, “Atomic Capabilities Layout,”](#) on page 1728.
- Added *set_deth_sqpn* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *deth_sqpn* to QP context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added *inner_first_svlan*, *inner_second_svlan*, *outer_first_svlan* and *outer_second_svlan* to [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- Renamed *vlan_tag* to *cvlan_tag* in [Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,”](#) on page 2005.
- Added *svlan_tag* in [Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,”](#) on page 2005.
- Updated descriptions of *first_prio*, *first_cfi* and *first_vid* in [Table 2125, “Flow Table Entry Match Set Layer 2-4 Field Description,”](#) on page 2006.

- Renamed *outer_second_vlan_tag* and *inner_second_vlan_tag* to *outer_second_cvlan_tag* and *inner_second_cvlan_tag* in Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Added *outer_second_svlan_tag* and *inner_second_svlan_tag* to Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009.
- Updated descriptions of *outer_second_prio*, *outer_second_cfi*, *outer_second_vid*, *inner_second_prio*, *inner_second_cfi* and *inner_second_vid* in Table 2131, “Flow Table Entry Match Set Misc Parameters Fields,” on page 2010.
- Updated Section 28.19.12, “PPCNT - Ports Performance Counters Register,” on page 1171
- Added Section 28.19.40, “PPLM - Port Phy Link Mode,” on page 1277
- Added Section 28.19.45, “SLTP - Serdes Lane Transmit Parameters,” on page 1302
- Added Section 28.19.47, “SLRG - Serdes Lane Receive Grade,” on page 1310

Revision 0.29 (November 2015)

- Renamed *pad_cap* to *end_pad* in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *start_pad* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *cache_line_128byte* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Updated Start of Packet Padding section. See Section 4.3.14, “Start of Packet Padding (RX),” on page 205.
- Updated End Of Packet Padding section. See Section 4.3.15, “End of Packet Padding (RX),” on page 205.
- Added Receive Data Segment section. See Section 8.9.4.1.16, “Receive Data Segments,” on page 341.
- Updated Send Data Segment section. See Section 8.9.4.1.15, “Send Data Segments,” on page 339.
- Added *self_lb_en* to MODIFY_TIR Bitmask. See Table 120, “MODIFY_TIR Bitmask,” on page 380.
- Added *self_lb_en_modifiable* to Per Protocol Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *retransmission_q_counters* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *duplicate_request*, *rnr_nack_retry_err*, *packet_seq_err*, *implied_nack_seq_err* and *local_ack_timeout_err* to Query Q Counter command. See Section 29.35, “Q Counters Commands,” on page 2185.

- Added *multi_path_qp* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *multi_path* to QP Context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *multi_path* to CREATE_QP required parameters. See Table 1885, “CREATE_QP Required Parameters,” on page 1902.
- Added *beacon_led* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added MLCR register. See Section 28.26.24, “MLCR – Management LED Control Register,” on page 1566.
- Added *rx_allow_untagged* to Roce Address. See Table 2532, “RoCE Address Field Descriptions,” on page 2174.
- Added *roce_rx_allow_untagged* to Roce Capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732.

Revision 0.28 (September 2015)

- Updated *wol_mode* and *rol_mode* to be bitmask instead of vector in SET_WOL_ROL command. See Table 1750, “SET_WOL_ROL Input Structure Layout,” on page 1846.
- Updated *wol_mode* and *rol_mode* to be bitmask instead of vector in SET_WOL_ROL command. See Table 1756, “QUERY_WOL_ROL Output Structure Layout,” on page 1848.
- Updated *wol_mode* and *rol_mode* to be bitmask instead of vector in ICM-D_SET_WOL_ROL command. See Table 2758, “ICMD_SET_WOL_ROL Input Mailbox Layout,” on page 2288.
- Updated *wol_mode* and *rol_mode* to be bitmask instead of vector in ICM-D_SET_WOL_ROL command. See Table 2760, “ICMD_QUERY_WOL_ROL Output Mailbox Layout,” on page 2289.
- Added SET_FLOW_TABLE_ROOT command. See Section 29.18.4, “SET_FLOW_TABLE_ROOT - Set Flow Table Root,” on page 1995.
- Added *modify_root* to Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753.
- Added Section 8.18.3.2, “Selecting the Root Flow Table,” on page 420.
- Added *eth_prio_primary_in_rts2rts* to Roce capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732.
- Added *eth_prio_of_primary_path* to RTS2RTS optional parameters. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.
- Added *driver_version* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Added `SET_DRIVER_VERSION` command. See [Section 29.3.19, “SET_DRIVER_VERSION,”](#) on page 1829.
- Updated driver start-up section to support setting of `driver_version`. See [Section 7.2, “HCA Driver Start-up,”](#) on page 244.
- Added `nic_vport_change_event` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Updated [Section 14.1.6, “NIC Vport Change Notification,”](#) on page 844.
- Added `source_sqn` to Flow Table Fields capabilities. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- Removed `source_sqn` limitation from Nic Transmit Flow Table. See [Section 8.18.4.4, “NIC Transmit,”](#) on page 429.
- Removed `source_sqn` limitation from E-switch Flow Table. See [Section 8.18.4.7, “E-Switch Forwarding Data Base \(FDB\),”](#) on page 430.
- Removed `source_sqn` limitation from Ingress ACL Flow Table. See [Section 8.18.4.8, “E-Switch Ingress Access Control List \(ACL\),”](#) on page 431.
- Removed `source_sqn` limitation from Egress ACL Flow Table. See [Section 8.18.4.9, “E-Switch Egress Access Control List \(ACL\),”](#) on page 432.
- Added `source_sqn` to [Table 2130, “Flow Table Entry Match Set Misc Parameters Format,”](#) on page 2009.
- Added `flow_tag` to limitations in Nic Transmit Flow Table. See [Section 8.18.4.4, “NIC Transmit,”](#) on page 429.
- Added `flow_tag` to limitations in Nic Transmit RDMA Flow Table. See [Section 8.18.4.5, “NIC Transmit RDMA,”](#) on page 429.
- Added `flow_tag` to limitations in E-switch Flow Table. See [Section 8.18.4.7, “E-Switch Forwarding Data Base \(FDB\),”](#) on page 430.
- Added `flow_tag` to limitations in Ingress ACL Flow Table. See [Section 8.18.4.8, “E-Switch Ingress Access Control List \(ACL\),”](#) on page 431.
- Added `flow_tag` to limitations in Egress ACL Flow Table. See [Section 8.18.4.9, “E-Switch Egress Access Control List \(ACL\),”](#) on page 432.
- Updated limitations of Nic Receive Flow Table to support `source_vport` for `vport_group_manager`. See [Section 8.18.4.1, “NIC Receive,”](#) on page 428.
- Added [Section 8.18.3.13.1, “Prio-Tagged and Untagged Packets,”](#) on page 428.
- Updated description of `bw_allocation` field in ETS tcN register. See [Table 250, “ETS tcN Configuration Register Fields,”](#) on page 712.
- Added [Section 8.18.2.9, “Flow Tagging,”](#) on page 418.
- Added `max_tc` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Updated [Section 8.18.3.5, “Allocating a Flow Counter,”](#) on page 422.

- Added *port_module_event* field to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Port Module Event. See Section 8.20.12.4, “Port Module Event,” on page 492.
- Removed *cq* from NODNIC Configuration Register. See Table 660, “NODNIC Configuration Registers Layout,” on page 815.
- Added *dc_connect_qp* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *multi_path_tir* to Flow Table Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.

Revision 0.27 (August 2015)

- Renamed *log_wqe_num_of_strides* to *single_wqe_log_num_of_strides* in WQ context. See Table 126, “Work Queue (WQ) Format,” on page 389.
- Renamed *log_wqe_stride_size* to *single_stride_log_num_of_bytes* in WQ context. See Table 126, “Work Queue (WQ) Format,” on page 389.
- Added *byte_cnt* to Error CQE. See Table 159, “Completion with Error CQE Layout,” on page 455.
- Added *wqe_inline_mode* to Network Offloading. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Renamed *self_lb_block_mc* and *self_lb_block_uc* to *self_lb_mc* and *self_lb_uc*. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Updated description of *self_lb_mc* and *self_lb_uc*. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added *min_wqe_inline_mode* to *field_select* field in Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.
- Updated *byte_cnt* description in CQE fields. See Table 147, “Completion Queue Entry Fields,” on page 437.
- Added HP OCBB and OCSD. See Section 6.8, “HP OCBB and OCSD,” on page 186.
- Added tool modification to WOL. See Section 7.5, “Wake On LAN (WOL),” on page 246.
- Added tool modification to ROL. See Section 7.6, “Reset On LAN (ROL),” on page 246.
- Added Non-Volatile Configuration chapter. See Chapter 11, “Non-Volatile Configuration (NVConfig)” (page 642).
- Added Non-Volatile Registers section. See Section 28.27, “Non-Volatile Registers,” on page 1571.
- Added ICMD Interface section. See Section , “,” on page 619.

- Added ICMD Command Reference. See [Chapter 30, “ICMD Command Reference”](#) (page 2267).
- Added Flow Counters support to Flow Table Chapter. See [Section 8.18, “Flow Table,”](#) on page 414.
- Added `ALLOC_FLOW_COUNTER` command. See [Section 29.18.12, “ALLOC_FLOW_COUNTER - Allocate a Flow Counter,”](#) on page 2038.
- Added `DEALLOC_FLOW_COUNTER` command. See [Section 29.18.13, “DEALLOC_FLOW_COUNTER - De-Allocate Flow Counter,”](#) on page 2040.
- Added `QUERY_FLOW_COUNTER` command. See [Section 29.18.14, “QUERY_FLOW_COUNTER – Query Flow Counter,”](#) on page 2041.
- Added *flow_counter_list_size* and *flow_counter_list* to Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,”](#) on page 2025.
- Added COUNT action to Flow Context. See [Table 2148, “FLOW_CONTEXT Structure Layout,”](#) on page 2025.
- Added Flow Counter entry. See [Table 2160, “Flow Counter List Entry Format Structure Layout,”](#) on page 2032.
- Added *max_flow_counter* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *flow_counter* and *log_max_flow_counter* to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Added *outer_ipv6_flow_label* and *inner_ipv6_flow_label* to Flow Table Fields. See [Table 1591, “Flow Table Fields Supported Format,”](#) on page 1761.
- Updated Packet Classification Ambiguities section. See [Section 8.18.2.7, “Packet Classification Ambiguities,”](#) on page 418.
- Added Resiliency and Error Recovery section to NO-DRAM NIC. See [Section 12.12, “Resiliency and Error Recovery,”](#) on page 829.
- Added Cqe Time Stamping section. See [Section 8.19.12, “CQE Timestamping,”](#) on page 471.
- Added *device_frequency* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *internal_timer_h* and *internal_timer_l* to Init segment. See [Table 14, “Initialization Segment,”](#) on page 234.
- Added Redefining Flow section. See [Section 8.18.3.10, “Redefining a Flow,”](#) on page 426.
- Added *modify_flow_en* to Flow Table Properties. See [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Added *modify_option* to `op_mod` field in Set Flow Table Command. See [Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,”](#) on page 2023.

- Added *modify_enable_mask* to Set Flow Table Command. See Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023.
- Added *cnp_prio_mode* to Congestion Control R-ROCE ECN. See Table 2673, “Congestion Control R-RoCE ECN NP Structure Layout,” on page 2241.
- Updated Eth Segment section. See Section 8.9.4.1.5, “Eth Segment and Padding Segment,” on page 318.
- Added Send Wqe Inline Header section. See Section 8.16.3, “Send WQE Inline Header,” on page 410.
- Added *min_wqe_inline_mode* to Nic Vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added *min_wqe_inline_mode* to SQ Context. See Table 138, “SQ Context Format,” on page 404.
- Removed *free_ar* from DCT context. See Table 25, “DCT Context Layout,” on page 279.
- Added SUSPEND_QP command. See Section 29.10.13, “SUSPEND_QP - Suspend a QP,” on page 1931.
- Added RESUME_QP command. See Section 29.10.14, “RESUME_QP - Resume a QP,” on page 1932.
- Added RDB Context. See Section 8.4, “RDB Context,” on page 278.
- Added SET_RDB command. See Section 29.10.15, “SET_RDB - SET RDB context,” on page 1933.
- Added QUERY_RDB command. See Section 29.10.16, “QUERY_RDB - QUERY RDB context,” on page 1935.
- Added *snapshot* field to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Snapshot Capabilities section. See Section 30.3.4.8, “Snapshot Capabilities,” on page 1344.
- Added *pre-init option* to *op_mod* in INIT_HCA command. See Table 1667, “INIT_HCA Input Structure Field Descriptions,” on page 1820.
- Added creation of *suspended qp option* to *op_mod* in CREATE_QP command. See Table 1882, “CREATE_QP Input Structure Field Descriptions,” on page 1900.
- Added *input_qpn* to CREATE_QP command. See Table 1882, “CREATE_QP Input Structure Field Descriptions,” on page 1900.
- Added *input_mkey_index* to CREATE_MKEY command. See Table 1798, “CREATE_MKEY Input Structure Field Descriptions,” on page 1862.
- Added *send_msg_psn*, *rcv_msg_psn*, *rdma_active*, *rdma_key*, *rdma_va*, *comm_est* and *suspended* fields to QP context. See Table 19, “QP Context Structure Layout,” on page 258.

Revision 0.26 (July 2015)

- Renamed *vsd* to *vlan_strip_disable* in RQ context. See [Table 124, “RQ Context Format,” on page 385.](#)
- Removed *user_index_equal_xrc_srqn* from XRC_SRQ context. See [Table 114, “XRC_SRQ Context Layout,” on page 370.](#)
- Updated [Section 5.3.2, “Gateway Ownership,” on page 233.](#)
- Made the following changes in Hca_vport_Context. See [Table 36, “HCA_Vport Context Layout,” on page 295:](#)
 - Renamed *port_state_policy* to *vport_state_policy*
 - Renamed *phy_port_state* to *port_state*
 - Added *port_physical_state*
 - Updated enumeration of *port_state* and *vport_state*
 - Added *system_image_guid*
- Added *port_num* to QUERY_HCA_VPORT_CONTEXT. See [Section 29.31.7, “QUERY_HCA_VPORT_CONTEXT – Query HCA Vport Context,” on page 2166.](#)
- Added *port_num* to MODIFY_HCA_VPORT_CONTEXT. See [Section 29.31.8, “MODIFY_HCA_VPORT_CONTEXT – Modify HCA Vport Context,” on page 2167.](#)
- Added *port_num* to QUERY_HCA_VPORT_GID. See [Section 29.31.9, “QUERY_HCA_VPORT_GID – Query HCA VPORT GID Table,” on page 2169.](#)
- Added *port_num* to QUERY_HCA_VPORT_PKEY. See [Section 29.31.10, “QUERY_HCA_VPORT_PKEY – Query HCA Vport Pkey Table,” on page 2170.](#)
- Added *port_num* to QUERY_VPORT_COUNTER. See [Section 29.34.2, “QUERY_VPORT_COUNTER – Query Vport Counter,” on page 2177.](#)
- Added *system_image_guid*, *port_guid*, *node_guid* and *qkey_violation_counter* to Nic_vport_Context. See [Table 29, “NIC_Vport Context Layout,” on page 287.](#)
- Added WOL section. See [Section 7.5, “Wake On LAN \(WOL\),” on page 246.](#)
- Added ROL section. See [Section 7.6, “Reset On LAN \(ROL\),” on page 246.](#)
- Added *rol_g*, *rol_s*, *wol_p*, *wol_u*, *wol_m*, *wol_b*, *wol_a*, *wol_g* and *wol_s* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Added SET_WOL_ROL command. See [Section 29.3.34, “SET_WOL_ROL – Set WOL and ROL Mode,” on page 1846.](#)
- Added QUERY_WOL_ROL command. See [Section 29.3.35, “QUERY_WOL_ROL – Query WOL and ROL Mode,” on page 1848.](#)
- Renamed *lro_max_ip_payload_size* to *lro_max_msg_sz* and updated description. See [Table 116, “TIR Context Format,” on page 374.](#)
- Updated description of *self_lb_block*. See [Table 116, “TIR Context Format,” on page 374.](#)

- Added *self_lb_block_mc* and *self_lb_block_uc* to Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734
- Added *lro_max_msg_sz_mode* to Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added Section 27.3, “Tracer,” on page 1051
- Added Tracer Registers. See Section 28.28, “Tracer Registers,” on page 1587.
- Added DC CNAK TRACE section. See Section 16.2.4, “DC CNAK Trace,” on page 874.
- Added SET_DC_CNAK_TRACE command. See Section 29.30.7, “SET_DC_CNAK_TRACE – Set DC CNAK Trace,” on page 2151.
- Added QUERY_DC_CNAK_TRACE command. See Section 29.30.8, “QUERY_DC_CNAK_TRACE – Query DC CNAK Trace,” on page 2153.
- Added *dc_cnak_trace* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added Nic Vport Change Notification section. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.
- Added Nic Vport Change Event. See Section 8.20.14.3, “NIC Vport Change Event,” on page 502.
- Added *promisc_uc*, *promisc_mc*, *promisc all* and *mtu* to nic vport context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added *arm_change_event*, *event_on_uc_address_change*, *event_on_mc_address_change*, *event_on_vlan_change*, *event_on_promisc_change* and *event_on_mtu* to Nic vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added *current_mc_mac_address* list to Nic vport Context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added new fields to *field_select* in MODIFY_NIC_VPORT_CONTEXT command. See Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.
- Added *ieee_vendor_id* to QUERY_ADAPTER command. See Table 1663, “QUERY_ADAPTER Parameters Block Layout,” on page 1818.
- Added *cq_period_start_from_cqe* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *cq_period_mode* to CQ context. See Table 171, “Completion Queue Context Layout,” on page 467.
- Updated Completion Event Moderation section to support *cq_period* upon completion. See Section 8.20.19, “Completion Event Moderation,” on page 508.

- Added Multi Packet Send Wqe Offload section. See [Section 4.3.4, “Multi-Packet Send WQE Offload,”](#) on page 191.
- Added *multi_pkt_send_wqe* to HCA_CAP. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Added *allow_multi_pkt_send_wqe* field to SQ context. See [Table 138, “SQ Context Format,”](#) on page 404.
- Updated description for fields in congestion parameters. See [Section 29.41, “Congestion Control Commands,”](#) on page 2228.
- Added *wqe_id* to Completion with error cqe. See [Table 159, “Completion with Error CQE Layout,”](#) on page 455.
- Added InfiniBand Virtualization section. See [Section 14.6, “InfiniBand Network Virtualization,”](#) on page 849.
- **Added CQE Compression section.** See [Section 8.19.9, “CQE Compressing,”](#) on page 459.
- Renamed *cqe_zip_en* to *cqe_compression_en* in CQ context. See [Table 171, “Completion Queue Context Layout,”](#) on page 467.
- Renamed *cqe_zip* to *cqe_compression* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Renamed *cqe_zip_timeout* to *cqe_compression_timeout* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Renamed *cqe_zip_max_num* to *cqe_compression_max_num* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Updated Characteristics of Flow Table types. See [Section 8.18.4.1, “NIC Receive,”](#) on page 428.
- Renamed *inline_scatter* to *cqe_format* in CQE format. See [Section 8.19.1.1, “CQE Format,”](#) on page 435.
- Fixed description for *lro_abrt* in CQE format. See [Table 147, “Completion Queue Entry Fields,”](#) on page 437.
- Updated description of *allowed_list_size* in Nic_vport_Context. See [Table 30, “NIC_Vport Context Field Descriptions,”](#) on page 289.
- Added *log_max_current_uc_list*, *log_max_current_mc_list* and *log_max_vlan_list* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *flow_tag* to [Table 1587, “Flow Table Properties Layout,”](#) on page 1753.
- Updated *flow_tag* description in [Table 2149, “FLOW_CONTEXT Field Descriptions,”](#) on page 2026.

Revision 0.25 (April 2015)

- Removed *user_index_equal_xrc_srqn* from XRC_SRQ context. See Table 114, “XRC_SRQ Context Layout,” on page 370.[Add to 0.26 rev history].
- Updated *user_index* in QP, DCT and XRC_SRQ. See Chapter 8, “Software Interface” (page 248),
- Update offset of *pas* in WQ context to 0xC0. See Table 126, “Work Queue (WQ) Format,” on page 389.
- Updated offset of *wq* in RMP context to 0x30. See Table 142, “RMP Context Format,” on page 412.
- Updated offset of *sq_context* to 0x20. See Table 2036, “QUERY_SQ Output Structure Layout,” on page 1965.
- Updated offset of *rq_context* to 0x20. See Table 2053, “QUERY_RQ Output Structure Layout,” on page 1971.
- Updated offset of *rmp_context* to 0x20. See Table 2095, “QUERY_RMP Output Structure Layout,” on page 1984.
- Updated offset of *rmp_context* in CREATE_RMP to 0x20. See Table 2081, “CREATE_RMP Input Structure Layout,” on page 1980.
- Updated offset of *rmp_context* in MODIFY_RMP to 0x20. See Table 2085, “MODIFY_RMP Input Structure Layout,” on page 1981.
- Deprecated modifying *srqn* in RTS2RTS command starting from ISSI==1. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.
- Updated *QPN/DCT/flow_tag* in CQE format to be valid for IB. See Table 147, “Completion Queue Entry Fields,” on page 437.
- Added *basic_cyclic_rcv_wqe* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *basic_cyclic_rcv_wqe* to RMP context. See Table 142, “RMP Context Format,” on page 412.
- Added *basic_cyclic_rcv_wqe* to XRC_SRQ context. See Table 114, “XRC_SRQ Context Layout,” on page 370
- Removed the deprecation of MAD_IFC from ISSI section. See Section 8.1.1.1, “ISSI = 1,” on page 249.
- Updated MAD_IFC command to be supported only for physical function drivers. See Section 29.24.3, “MAD_IFC - Management Datagram Interface,” on page 2101.
- Removed *rsz_srqn* and *log_max_strq_sz* from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *log_max_wq_sz* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added XRC_SRQ commands. See Section 29.26, “XRC_SRQ Commands,” on page 2109.

- Added deprecation for SRQ commands from `ISSI==1`. See [Section 8.1.1.1, “ISSI = 1,”](#) on [page 249](#).
- Added `XRC_SRQ` section. See [Section 8.11, “XRC Shared Receive Queues,”](#) on [page 370](#).
- Removed `user_index` from SRQ context. See [Table 112, “SRQ Context Layout,”](#) on [page 367](#).
- Removed `phy_port_owner` from the following:
 - [Table 1571, “HCA Capabilities Layout,”](#) on [page 1671](#)
 - [Section 3.2.10, “Port Control Registers,”](#) on [page 183](#)
 - [Section 19.3, “QoS Control Registers,”](#) on [page 930](#)
- Removed `cd_slave` from [Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,”](#) on [page 395](#)
- Added `rmppn` to QP context. See [Table 19, “QP Context Structure Layout,”](#) on [page 258](#).
- Added `single_mac` field to [Table 964, “PPAD - Port Physical Address Register Layout,”](#) on [page 1157](#).
- Updated [Section 28.19.11, “PFCC - Ports Flow Control Configuration Register,”](#) on [page 1165](#).
- Removed `user_index` from QP Event. See [Table 180, “Event_data Field - QP/SQ/RQ Events Layout,”](#) on [page 481](#).
- Added `rqn`, `sqn` and `type` to QP Event. See [Table 180, “Event_data Field - QP/SQ/RQ Events Layout,”](#) on [page 481](#).
- Updated SQ states figure. See [Figure 26, “SQ States,”](#) on [page 403](#).
- Added RST state to state in SQ context. See [Table 139, “SQ Context Fields,”](#) on [page 405](#).
- Added RST2RDY and 2RST CREATE_SQ and MODIFY_SQ Bitmask. See [Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,”](#) on [page 407](#).
- Updated SQ States Summary. See [Table 141, “SQ States Summary,”](#) on [page 409](#).
- Removed state from TIS context. See [Table 121, “TIS Context Format,”](#) on [page 382](#).
- Removed `tis_state` from `Modify_TIS`.
- Updated MODIFY TIS BITMASK. See [Table 123, “MODIFY_TIS Bitmask,”](#) on [page 383](#).
- Updated RQ States figure. See [Figure 25, “RQ States,”](#) on [page 384](#).
- Added RST state to state in RQ context. See [Table 125, “RQ Context Fields,”](#) on [page 386](#).
- Added RST2RDY and 2RST CREATE_RQ and MODIFY_RQ Bitmask. See [Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,”](#) on [page 395](#).
- Updated WQE Ownership section. See [Section 8.9.3, “WQE Ownership,”](#) on [page 309](#).
- Added `port_type` to `HCA_CAP`. See [Table 1571, “HCA Capabilities Layout,”](#) on [page 1671](#).
- Removed `choose_current_uc_perm` from Nic Vport Context. See [Table 29, “NIC_Vport Context Layout,”](#) on [page 287](#).
- Updated description of `allowed_list_size` in Nic Vport Context. See [Table 29, “NIC_Vport Context Layout,”](#) on [page 287](#).

- Removed *choose_current_uc_perm* from *field_select* in Modify Nic Vport Context. See Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.
- Removed *flow_tag* in CQE format from SQ column.
- Added *compact_address_vector* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Updated bit 0 of op_mod in SET_HCA_CAP to be reserved. See Section 29.3.5, “SET_HCA_CAP – Set Device Capabilities,” on page 1803.
- Replaced *XOR* with *Inverted Xor8* for *rx_hash_fn* in TIR context. See Table 116, “TIR Context Format,” on page 374
- Renamed *ethernet* to *eth_net_offloads* in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Renamed 1 to *ext* in UD Address Vector. See Table 109, “UD Address Vector Layout,” on page 362
- Added *user_index* to Create QP Required Parameters. See Table 1885, “CREATE_QP Required Parameters,” on page 1902.
- Renamed *indirect_tlb* to *indirect_table* in TIR context. See Table 116, “TIR Context Format,” on page 374.
- Added *indirect_table* as optional parameter in MODIFY_TIR. See Table 120, “MODIFY_TIR Bitmask,” on page 380.
- Added *vport_counters* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Rename *packet_len* to *len* in WQE Associated Page Fault Event. See Table 190, “Event_data Field - WQE Associated Page Fault Event Field Descriptions,” on page 484.
- Updated offset of congestion parameters to be in 0x20 to 0x11c instead of 0x20 to 0x12c. See Table 2675, “MODIFY_CONG_PARAMS Input Structure Layout,” on page 2243.
- Added *esw_vport_context*. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.
- Updated *query_esw_vport* command. See Section 29.31.3, “QUERY_ESW_VPORT_CONTEXT – Query E-switch Vport Context,” on page 2158.
- Updated *modify_esw_vport* command. See Section 29.31.4, “MODIFY_ESW_VPORT_CONTEXT – Modify E-switch Vport Context,” on page 2160
- Added E-switch capabilities to op_mode in QUERY_HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *eswitch_flow_table* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768.

- Added *flow_table_properties_nic_receive_sniffer* and *flow_table_properties_nic_transmit_sniffer* to Flow Table Nic capabilities, See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
- Added Exponential Backoff Time section to Network/Transport Enhancements, See Section 13.2.5, “Exponential Backoff Timer,” on page 832
- Added log_rtm to Address Path, See Table 21, “Address Path Structure Layout,” on page 274.
- Added log_rtm to required parameters in RTR2RTS, See Table 1910, “RTR2RTS_QP Required Parameters,” on page 1916.
- Added log_rtm to optional parameters in INIT2RTR, RTR2RTS and RTS2RTS.
- Added exponential_backoff to HCA_CAP, See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Rewrote PCI Interface section. See Chapter 5, “PCI Interface” (page 226).
- Added PCI vendor-specific capability documentation. See Section 5.3, “VSC - Vendor Specific Capability,” on page 227.
- Added *nic_interface*, *nic_interface_supported*, *no_dram_nic_offset*, *clear_int* to initialization segment. See Section 5.4, “Initialization Segment,” on page 234.
- Removed *clear_intx* from initialization segment. See Section 5.4, “Initialization Segment,” on page 234.
- Added NO-DRAM NIC chapter. See Chapter 12, “No-DRAM NIC” (page 813)
- Removed *intapin* from QUERY_ADAPTER. See Section 29.3.12, “QUERY_ADAPTER – Query Adapter,” on page 1816
- Updated HCA Driver Startup section. See Section 7.2, “HCA Driver Start-up,” on page 244.

Revision 0.24 (February 2015)

- Removed *inner_address* from nic_vport context. See Table 29, “NIC_Vport Context Layout,” on page 287.
- Added L2_Table commands. See Section 29.19, “L2 Table Commands,” on page 2058.
- Updated Driver Start-up section. See Section 7.2, “HCA Driver Start-up,” on page 244.
- Added *log_max_l2_table* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *log_max_rmp* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Removed *group_id* from SET_FLOW_TABLE_ENTRY. See Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023.
- Added *group_id* to FLOW Table Entry Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.
- Added *flow_tag* to FLOW Table Entry Context. See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025.

- Moved *table_id* in CREATE_FLOW_TABLE output command to 0x8 instead of 0xC. See Table 2100, “CREATE_FLOW_TABLE Output Structure Layout,” on page 1988.
- Added description for *tcp_flags* in FLOW TABLE ENTRY. See Table 2125, “Flow Table Entry Match Set Layer 2-4 Field Description,” on page 2006.
- **Changed Ethernet and Roce vport counters offsets.** See Section 29.34, “Vport Counters-Monitoring Commands,” on page 2176
- Updated *log_max_sq* in HCA_CAP to be RW. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Updated Modify RQ bitmask so that rmpn and cqn cannot be modified. See Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395.
- Updated Modify SQ bitmask so that cqn cannot be modified. See Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407.
- Removed STRQ from *wq_type* in WQ Context. See Table 127, “Work Queue (WQ) Fields,” on page 391.
- Removed STRQ from *rq_type* in QP context. See Table 20, “QP Context Field Descriptions,” on page 261.
- Moved *log_wq_pg_sz*, *log_wq_stride* and *log_wq_size* in WQ context to offset 0x20h. See Table 126, “Work Queue (WQ) Format,” on page 389.
- Added *log_min_stride_sz_rq*, *log_min_stride_sz_rq*, *log_min_stride_sz_sq*, *log_min_stride_sz_sq* to HCA CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Removed *cs* from RQ context. See Table 124, “RQ Context Format,” on page 385.
- Updated *hw_rq_counter* and *sw_rq_counter* to be 32 bits with reserved 16 msb bits in QP context. See Table 20, “QP Context Field Descriptions,” on page 261.
- Renamed *sw_context* to *user_index* in CQE format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435
- Renamed *sw_context* to *user_index* in QP Event. See Table 180, “Event_data Field - QP/SQ/RQ Events Layout,” on page 481.
- Renamed *sw_context* to *user_index* in DCT context. See Table 25, “DCT Context Layout,” on page 279.
- Renamed *sw_context* to *user_index* in RQ context. See Table 124, “RQ Context Format,” on page 385.
- Renamed *sw_context* to *user_index* in SQ context. See Table 138, “SQ Context Format,” on page 404.
- Added *user_index* to QP context. See Table 19, “QP Context Structure Layout,” on page 258.
- Added *user_index* to SRQ context. See Table 112, “SRQ Context Layout,” on page 367.

- Added *cqe_version* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Removed *sniffer_rule_phy*, *sniffer_rule_vport* and *sniffer_rule_flow* from HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Removed CREATE_SNIFFER_RULE and DESTROY_SNIFFER_RULE commands.
- Added *counter_set_id* to RQ context. See [Table 124, “RQ Context Format,” on page 385.](#)
- Added nic sniffer rx and nic sniffer tx to *table_type* in CREATE_FLOW_TABLE command. See [Table 2099, “CREATE_FLOW_TABLE - Input Structure Field Description,” on page 1987.](#)
- Added nic sniffer rx and nic sniffer tx to *table_type* in CREATE_FLOW_GROUP command. See [Table 2121, “CREATE_FLOW_GROUP - Input Structure Field Description,” on page 2001.](#)
- Added nic sniffer rx and nic sniffer tx to *table_type* in CREATE_FLOW_TABLE_ENTRY command. See [Table 2147, “SET_FLOW_TABLE_ENTRY - Input Structure Field Descriptions,” on page 2023.](#)
- Added Sniffer section. See [Section 4.5, “Sniffer,” on page 207.](#)
- Removed MLX QP and MLX WQE segment.
- Removed *xrc_odp_caps* and *dc_odp_caps* from [Table 1573, “ODP Capabilities Layout,” on page 1726.](#)
- Added *out_of_sequence* counter to Q counters. See [Section 29.35, “Q Counters Commands,” on page 2185.](#)
- Added *out_of_seq_cnt* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Added QUERY_FLOW_TABLE command. See [Section 29.18.5, “QUERY_FLOW_TABLE - Query Flow Table,” on page 1998.](#)
- Added QUERY_FLOW_GROUP command. See [Section 29.18.8, “QUERY_FLOW_GROUP - Query Flow Group,” on page 2020.](#)
- Added *log_max_transport_domain* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Added *transport_domain* and *self_lb_block* to TIR Context. See [Table 116, “TIR Context Format,” on page 374.](#)
- Added *transport_domain* to TIS Context. See [Table 121, “TIS Context Format,” on page 382.](#)
- Added ALLOC_TRANSPORT_DOMAIN command. See [Section 29.3.32, “ALLOC_TRANSPORT_DOMAIN - Allocate Transport Domain,” on page 1844.](#)
- Added DEALLOC_TRANSPORT_DOMAIN command. See [Section 29.3.33, “DEALLOC_TRANSPORT_DOMAIN - De-Allocate Transport Domain,” on page 1845.](#)

- Added Self-Loopback control section. See [Section 4.4, “Self-Loopback Control Using Transport Domains,”](#) on page 206.
- Removed *log_max_rql* and *log_max_rq_per_rql* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_rqt* and *log_max_rqt_size* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Removed RQL context.
- Added RQT context. See [Section 8.15, “RQ Table \(RQT\),”](#) on page 399.
- Removed RQL command.
- Added RQT command. See [Section 29.16, “RQT Commands,”](#) on page 1974.
- Update *disp_type* in TIR Context. See [Table 116, “TIR Context Format,”](#) on page 374.
- Removed *rql_number* from TIR context. See [Table 116, “TIR Context Format,”](#) on page 374.
- Added *inline_rqn* to TIR context. See [Table 116, “TIR Context Format,”](#) on page 374.
- Updated RSS section. See [Section 4.3.6, “Receive Side Scaling \(RSS\),”](#) on page 194.
- Removed profile from INIT_HCA. See [Table 1666, “INIT_HCA Input Structure Layout,”](#) on page 1820.
- Removed ACTIVATE_TRACER and DEACTIVATE_TRACER commands.
- Added *Ethernet* and *pad_tx_eth_packet* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Update flow table entry section. See [Section 8.18, “Flow Table,”](#) on page 414.
- Renamed *flow_table* to *nic_flow_table* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Updated Flow Table capabilities. See [Section 29.3.4.6, “Flow Table Capabilities,”](#) on page 1741.
- Updated Flow Table Commands. See [Section 29.18, “Flow Table Commands,”](#) on page 1985.
- Removed optional parameters from SQD2RTS command. See [Section 29.10.12, “SQD2RTS_QP – SQD to RTS QP,”](#) on page 1929

Revision 0.23 (January 2015)

- Added *drain_sigerr* to QP context. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Added *drain_sigerr* to HCA_CAP. See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on page 1667.
- Supported draining when signature error occurs. See [Section 17.3.14.2, “Signature Handover Completion Report,”](#) on page 895.

- Added SQD2RTS command. See [Section 29.10.12, “SQD2RTS_QP – SQD to RTS QP,”](#) on page 1929.
- Added *flush_in_error_en* to RQ context. See [Section 8.14, “Receive Queue \(RQ\),”](#) on page 383.
- Added RQ Error Semantics section. See [Section 8.14.2, “RQ Error Semantics,”](#) on page 398.
- Added *flush_in_error_en* to SQ context. See [Section 8.16, “Send Queue \(SQ\),”](#) on page 402.
- Added SQ Error Semantics section. See [Section 8.16.2, “SQ Error Semantics,”](#) on page 409.
- Renamed ‘SRQ Last WQE Reached Event’ to Last WQE Reached Event. See [Section 8.20.6, “Event Queue Mapping,”](#) on page 476.
- Added CQ to EQ Remapping section. See [Section 8.19.11, “CQ to EQ Remapping,”](#) on page 471.
- Added *cq_eq_remap* to HCA_CAP. See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on page 1667.
- Added *c_eqn* to CQ Modify Field Select. See [Table 1875, “MODIFY_FIELD_SELECT Structure Field Descriptions,”](#) on page 1896.
- Added *scqe_break_moderation* to HCA_CAP. See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on page 1667.
- Added *scqe_break_moderation_en* to CQ context. See [Section 8.19.10, “Completion Queue Context \(CQC\),”](#) on page 689.
- Added *rql_max_size* to RQL context. See [Section 8.15, “RQ Table \(RQT\),”](#) on page 399.
- Renamed *rq_index* to *sw_context* in RQ context. See [Table 124, “RQ Context Format,”](#) on page 385.
- Renamed *sq_index* to *sw_context* in SQ context. See [Table 138, “SQ Context Format,”](#) on page 404.
- Added *udp_sport* to RTS2RTS_QP optional parameters mask. See [Table 1916, “RTS2RTS_QP Optional Parameters Mask,”](#) on page 1919.
- Added *inner_address* to Nic_vport Context. See [Table 29, “NIC_Vport Context Layout,”](#) on page 287.
- Removed *flow_table_field_support*, *log_flow_table_sz*, *flow_table_field_bitmask_support* and *eswitch_forwarding_tcam_field_support* from [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Added *flow_table* to HCA_CAP. See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on page 1667.
- Added Flow Table Capabilities to HCA_CAP. See [Section 21.3.3.6, “Flow Table Capabilities,”](#) on page 472.

- Removed connector field from RQ context. See [Section 8.14, “Receive Queue \(RQ\),” on page 383.](#)
- Added ethernet field to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Updated description of *allowed_list_size* field in Nic_vport Context. See [Table 30, “NIC_Vport Context Field Descriptions,” on page 289.](#)
- Added bit 3: permanent_address to field_select field in [Table 2508, “MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions,” on page 2164.](#)

Revision 0.22 (December 2014)

- Removed *rx_hash_inner* from TIR context. See [Table 116, “TIR Context Format,” on page 374.](#)
- Updated ODP section to support SRQ. See [Section 9.1.2, “On-demand Paging Support,” on page 626.](#)
- Added *xrc_odp_caps* to ODP capabilities. See [Table 1573, “ODP Capabilities Layout,” on page 1726.](#)
- Added *dc_odp_caps* to ODP capabilities. See [Table 1573, “ODP Capabilities Layout,” on page 1726.](#)
- Added *srq* and *read* capability to ODP per transport capabilities. See [Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727.](#)
- Added *xrc_srq* and *dct* to WQE associated page fault event. See [Table 189, “Event_data Field - WQE Associated Page Fault Event Layout,” on page 484.](#)
- Marked *cd_slave* as valid for RQ in WQ. See [Table 127, “Work Queue \(WQ\) Fields,” on page 391.](#)
- Updated offset of *hca_vport_context* in *modify_hca_vport_context* to 0x10. See [Section 29.31.8, “MODIFY_HCA_VPORT_CONTEXT – Modify HCA Vport Context,” on page 2167.](#)
- Renamed *log_max_rq_per_tir* to *log_max_rq_per_rql* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Added *roce_apm* to ROCE capabilities. See [Table 1579, “RoCE Capabilities Layout,” on page 1732.](#)
- Updated Cqe Checksum Behavior. See [Table 6, “CQE Checksum Behavior,” on page 190.](#)
- Added Ethernet Virtualization section. See [Section 14, “Virtualization,” on page 838](#)
- Updated the initializing stage behavior in HCA_DRIVER_STARTUP. See [Section 7.2, “HCA Driver Start-up,” on page 244.](#)
- Updated Nic_vport Context. See [Table 29, “NIC_Vport Context Layout,” on page 287.](#)

- Removed Esw_Vport Context from [Chapter 8, “Software Interface”](#) (page 248).
- Removed FDB Commands
- Made several changes to [Table 541, “SET_FLOW_TABLE_ENTRY Input Structure Layout,”](#) on page 472:
 - Added action to offset 0x10 bits 27:24
 - Updated destination_id field to destination_id/num_dst_vport
 - Added table_name to offset 0x14 bits 27:24
 - Added dst_vport to offset 0x3A0 bits 15:0
- Added *eswitch_forwarding_tcam_field_support* to [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.

Revision 0.21 (December 2014)

- **[Shomron only]** Added FDB commands. See [Section 21.21, “FDB Commands,”](#) on page 607.
- **[Shomron only]** Increased size of *lro_timeout_period_usecs* in TIR context to 16 bits. See [Table 116, “TIR Context Format,”](#) on page 374.
- **Added roce counters to VPORT_COUNTERS.** See [Section 29.34.2, “QUERY_VPORT_COUNTER – Query Vport Counter,”](#) on page 2177.
- Added ISSI mechanism. See [Section 8.1, “ISSI - Interface Step Sequence ID,”](#) on page 248.
- Updated HCA Driver Startup to include QUERY_ISSI and SET_ISSI. See [Section 7.2, “HCA Driver Start-up,”](#) on page 244.
- Added QUERY_ISSI command. See [Section 29.3.17, “QUERY_ISSI,”](#) on page 1826.
- Added SET_ISSI command. See [Section 29.3.18, “SET_ISSI,”](#) on page 1828.
- Deprecated MAD_IFC command starting from ISSI =1. See [Section 29.24.3, “MAD_IFC - Management Datagram Interface,”](#) on page 2101.
- Made the following changes in HCA_CAP (see [Table 1571, “HCA Capabilities Layout,”](#) on page 1671):
 - Renamed *eswitch_owner* to *vport_group_manager*
 - Added *ib_virt* to HCA_CAP
 - Added *eth_virt* to HCA_CAP
 - Added *gid_table_size* to HCA_CAP
- Updated ENABLE_HCA command to support virtualization. See [Section 29.3.15, “ENABLE_HCA,”](#) on page 1823.
- Updated DISABLE_HCA command to support virtualization. See [Section 29.3.16, “DISABLE_HCA,”](#) on page 1825.

- Updated *vsd*, *cqn* and *rmprn* fields in Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395
- Updated *cqn* field in Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395
- Added QUERY_HCA_VPORT_CONTEXT. See Section 29.31.7, “QUERY_HCA_VPORT_CONTEXT – Query HCA Vport Context,” on page 2166.
- Added MODIFY_HCA_VPORT_CONTEXT. See Section 29.31.8, “MODIFY_HCA_VPORT_CONTEXT – Modify HCA Vport Context,” on page 2167.
- Added QUERY_HCA_VPORT_GID command. See Section 29.31.9, “QUERY_HCA_VPORT_GID – Query HCA VPORT GID Table,” on page 2169.
- Added QUERY_HCA_VPORT_PKEY Command. See Section 29.31.10, “QUERY_HCA_VPORT_PKEY – Query HCA Vport Pkey Table,” on page 2170.
- Added HCA_Vport Context. See Section 8.8, “HCA_Vport Context - HCA Virtual Port Context,” on page 295.
- Added function_id to QUERY_PAGES command. See Section 29.3.1, “QUERY_PAGES - Query Device Free Memory Pool Status,” on page 1660.
- Added Hierarchical Structure of Flow Tables, TIRs and RQs section. See Section 4.2, “Hierarchical Structure Of Flow Tables,TIRs and RQs,” on page 107.
- Added Ethernet Multicast section. See Section 4.3, “Stateless Offloads,” on page 188.
- Removed *rq_type* from RQ context. See Section 8.14, “Receive Queue (RQ),” on page 383.
- Added *mem_rq_type* and connector to RQ context. See Section 8.14, “Receive Queue (RQ),” on page 383.
- Added *log_max_rql* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added RQL context. See Section 8.15, “RQ Table (RQT),” on page 399.
- Added RQL commands. See Section 29.16, “RQT Commands,” on page 1974.
- Removed *rq_lst_sz*, *rx_hash_default_rqn*, *rx_hash_field_selector_ipv4*, *rx_hash_field_selector_ipv6* and *rq_num* from TIR context. See Section 8.12, “Transport Interface Receive (TIR),” on page 373.
- Added *rql_number*, *rx_hash_fields_selector_outer* and *rx_hash_fields_selector_inner* to TIR context. See Section 8.12, “Transport Interface Receive (TIR),” on page 373.
- Updated TIR commands size. See Section 29.12, “TIR Commands,” on page 1948.
- Updated Receive Side Scaling section. See Section 4.3.6, “Receive Side Scaling (RSS),” on page 194.
- Added RSS Example. See Section 4.4.3.2, “RSS Example,” on page 91.
- Added *ets* field to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added ETS registers. See Section 19.3, “QoS Control Registers,” on page 930.

- Added *tunnel_lro_gre*, *tunnel_lro_vxlan*, *tunnel_stateless_gre* and *tunnel_stateless_vxlan* to Per Protocol Networking Offloads Capabilities, See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Removed Networking Offloads Capabilities table from HCA_CAP. See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on page 1667.
- Renamed Networking Offloads Capabilities in op_mod of QUERY_HCA_CAP command to Ethernet Offloads Capabilities. See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667.
- Added IPoIB Offloads Capabilities and EoIB Offloads Capabilities to QUERY_HCA_CAP command to Ethernet Offloads Capabilities, See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667.
- Resized Per Protocol Networking Offloads Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Added *tunnel_iso_const_outer_ip_id* to Per Protocol Networking Offloads Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.
- Added VXLAN Commands. See [Section 29.42, “VXLAN Commands,”](#) on page 2251.
- Added *tunneled_offload_en* to TIR context. See [Table 116, “TIR Context Format,”](#) on page 374.
- Added Hypervisor Offloading for Tunneling section. See [Section 4.3.12, “Hypervisor Offloading for Tunneling,”](#) on page 201.
- Updated Checksum Offloads section. See [Section 4.3.1, “Checksum Offload,”](#) on page 189.
- Updated Vlan insertion/stripping section. See [Section 4.3.10, “VLAN Insertion/Stripping,”](#) on page 201.
- Updated Cqe Layout Format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
- Added QUERY_CONG_STATUS command. See [Section 29.41.1, “QUERY_CONG_STATUS - Query Congestion Status,”](#) on page 2228
- Added MODIFY_CONG_STATUS command. See [Section 29.41.2, “MODIFY_CONG_STATUS - Query Congestion Status,”](#) on page 2230
- Removed enable fields from Congestion protocols structs. See [Section 29.41, “Congestion Control Commands,”](#) on page 2228.
- Removed *cur_total_rate*, *sum_total_rate*, *min_total_rate*, *max_total_rate* from. See [Table 2687, “QUERY_CONG_STATISTICS Output Structure Layout,”](#) on page 2247.
- Added fc_proto_admin, fc_proto_oper, eth_proto_lp_advertise to [Table 246, “PTYS - Port Type and Speed Register Layout,”](#) on page 409.
- Updated [Table 254, “PFCC - Ports Flow Control Configuration Register Layout,”](#) on page 415.
- Added pnat to [Table 256, “PPCNT - Ports Performance Counters Register Layout,”](#) on page 416

- Added [Table 20.5.9.11, “Physical Layer Counters Group,”](#) on page 441
- Removed Allowed Address context. See [Table 38, “Allowed Address Context Layout,”](#) on page 143.
- Added [Section 28.19.38, “PTER - Port Transmit Errors Register,”](#) on page 1270
- Added [Section 28.19.40, “PPLM - Port Phy Link Mode,”](#) on page 1277

Revision 0.20 (September 2014)

- Updated layout of Allowed Address context. See [Table 38, “Allowed Address Context Layout,”](#) on page 143.
- Updated layout of Address context. See [Table 40, “AddressMac Address Context Layout,”](#) on page 146.
- Added Roce Vlan Behavior table. See [Table 2533, “RoCE VLAN Behavior,”](#) on page 2174.
- Removed SQD state from Qp context. See [Table 20, “QP Context Field Descriptions,”](#) on page 261.
- Renamed *f_eup* to *f_eth_prio*. See [Table 21, “Address Path Structure Layout,”](#) on page 274.
- Renamed *prio* to *eth_prio*. See [Table 21, “Address Path Structure Layout,”](#) on page 274.
- Added LRO Cqe fields summary. See [Section 4.3.9.6, “LRO CQE Fields Summary,”](#) on page 198.
- Removed valid bit from RoCE Address layout. See [Table 2531, “RoCE Address Layout,”](#) on page 2173.
- Renamed *lro_max_merged_segment_size* to *lro_max_ip_payload_size* in TIR context. See [Table 116, “TIR Context Format,”](#) on page 374.
- Updated Vport Counters. See [Section 29.31, “Vport Commands,”](#) on page 2154.
- Renamed Received/ Transmitted multicast to Received/Transmitted broadcast in *Query_Vport_Counters*. See [Table 2542, “QUERY_VPORT_COUNTER Output Structure Layout,”](#) on page 2178.
- Added Cqe Zipping. See [Section 8.19.9, “CQE Compressing,”](#) on page 682
- Added Cqe Zipping capabilities to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *cqe_zip_en* to CQ context. See [Table 171, “Completion Queue Context Layout,”](#) on page 467.
- Added *mini_cqe_res_format* to CQ context. See [Table 171, “Completion Queue Context Layout,”](#) on page 467.
- Added Aggregated cqe type to inline_scatter in CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.

- Added *mini_cqe_num* to CQE format. See [Table 145, “64B Completion Queue Entry Format Layout,”](#) on page 435.
- Removed *f_grh_tclass* from Address Path. See [Table 21, “Address Path Structure Layout,”](#) on page 274
- Added Roce required field for DCT context. See [Table 25, “DCT Context Layout,”](#) on page 279.
- Renamed *src_addr_index* to *my_addr_index* in DCT context. See [Table 25, “DCT Context Layout,”](#) on page 279.
- Added Congestion Control section. See [Section 3.2.4, “Congestion Control,”](#) on page 182.
- Added Congestion Control commands. See [Section 29.41, “Congestion Control Commands,”](#) on page 2228.
- Added *cc_query_allowed* and *cc_modify_allowed* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added port control registers. See [Section 28.19, “Network Ports Registers,”](#) on page 1140.

Revision 0.19 (August 2014)

- Added [Table 972, “TRAFFIC_COUNTER Format Layout,”](#) on page 738
- Updated SET_FLOW_TABLE_ENTRY command, See [Table 541, “SET_FLOW_TABLE_ENTRY Input Structure Layout,”](#) on page 472.
- Renamed *lro_timeout_period_index* to be *lro_timeout_period_usecs* in TIR context, See [Table 116, “TIR Context Format,”](#) on page 374.
- Added *log_max_tis* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_tir* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_rq* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_sq* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_rq_per_tir* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *log_max_tis_per_sq* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.
- Added *roce_address_table_size* to roce capabilities. See [Table 1579, “RoCE Capabilities Layout,”](#) on page 1732.
- Removed *log_max_tis* from Networking Offload Capabilities. See [Table 1581, “Per Protocol Networking Offload Capabilities Layout,”](#) on page 1734.

- Removed *log_max_tir* from Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Removed *multiple_tis_per_sq* from Networking Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
- Added QUERY_ROCE_ADDRESS command. See Section 29.27.5, “QUERY_XRQ_DC_PARAMS_ENTRY – Query XRQ DC PARAMS Entry,” on page 2122.
- Added SET_ROCE_ADDRESS command. See Section 29.33, “SET_ROCE_ADDRESS – SET RoCE Address Entry,” on page 2175.
- Added valid to Roce Address, See Table 2531, “RoCE Address Layout,” on page 2173
- Removed *st* and *state* from TIR context. See Table 116, “TIR Context Format,” on page 374.
- Moved *rq_list* to offset 0xf0 in TIR context, See Table 116, “TIR Context Format,” on page 374.
- Removed *st* and *disp_type* from create_tir, See Table 1988, “CREATE_TIR Input Structure Layout,” on page 1949.
- Removed *st* from create_tis, See Table 2005, “CREATE_TIS Input Structure Layout,” on page 1955.
- Removed ERR state from TIS. See Table 2005, “CREATE_TIS Input Structure Layout,” on page 1955.
- Removed *tir_state* from TIR Commands. See Table 29.12, “TIR Commands,” on page 1948.
- Removed *st* from TIS context. See Table 121, “TIS Context Format,” on page 382.
- Swapped *wq_type* fields in wq context. See Table 127, “Work Queue (WQ) Fields,” on page 391
- Added *roce_en* to nic_vport context. See Section 29.31, “Vport Commands,” on page 2154.
- Added other_vport to Vport commands, See Section 29.31, “Vport Commands,” on page 2154.
- Added other_vport to QUERY_Vport_COUNTER command. See Section 29.31, “Vport Commands,” on page 2154.
- Renamed *esw_modify_allowed* to *eswitch_owner*. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Removed *ptp1588* from *st* in QP context. See Table 20, “QP Context Field Descriptions,” on page 261.
- Removed *sniffer_physical_port* from HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.
- Added *sniffer_rule_phy* to HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.

- Added *sniffer_rule_vport* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Added *sniffer_rule_flow* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671.](#)
- Removed *sniffer* from *st* in QP context. See [Table 20, “QP Context Field Descriptions,” on page 261.](#)
- Updated *tir_context* at 0x20. See [Table 2003, “QUERY_TIR Output Structure Field Descriptions,” on page 1954.](#)

Revision 0.18 (June 17, 2014)

- Changed SQ to point to TIS (previously was TIS pointed to SQ)
 - Removed from TIS: *sq_lst_sz*, *sq_num[i]* see [Section 4.1.4, “Transport Interface Send \(TIS\),” on page 84](#)
 - Added *tis_lst_sz* and *tis_num[0]* to SQ, see [Section 4.1.6, “Send Queue \(SQ\),” on page 93](#)
 - Updated [Figure 11, “Networking Transport Objects,” on page 186](#)
 - Added *multiple_tis_per_sq* to QUERY_HCA_CAP, see [Section 22.1, “Networking Offload Capabilities,” on page 498](#)
- Clarified TIR fields relevant for *rx_hash_fn* =HASH_NONE: *rx_hash_symmetric*, *rx_hash_inner*, *rx_hash_field_selector_ipv4*, *rx_hash_field_selector_ipv6*
- Added LRO
 - Added relevant capabilities in QUERY_HCA_CAP (*lro_cap*, *lro_psh_flag*, *lro_time_stamp*, *lro_min_mss_size*, *lro_timer_supported_periods*, *lro_min_mss_size*, *lro_timer_supported_periods[4]*), see [Section 22.1, “Networking Offload Capabilities,” on page 498](#)
 - Updated TIR LRO related fields *lro_timeout_period_index*, *lro_enable_mask*, *lro_max_merged_segment_size*, see [Section 4.1.3, “Transport Interface Receive \(TIR\),” on page 78](#)
 - Created overview section for LRO, see [Section 4.3.9, “Large Receive Offload \(LRO\),” on page 195](#)
- Added Roce Address entry. [Table 2531, “RoCE Address Layout,” on page 2173.](#)
- Updated QUERY_NIC_Vport_CONTEXT and MODIFY_NIC_Vport_CONTEXT to support roce.
- Removed *link_type* from Qp.address_path context, See [Table 21, “Address Path Structure Layout,” on page 274.](#)
- Removed *link_type* from DCT context, See [Table 25, “DCT Context Layout,” on page 279.](#)
- Removed *vsd* and *rme* from Qp context, See [Table 19, “QP Context Structure Layout,” on page 258](#)
- Renamed *mgid_index* to *src_addr_index* in Qp.address_path context, See [Table 21, “Address Path Structure Layout,” on page 274.](#)

- Updated Qp.address_path to support Roce, See [Table 21, “Address Path Structure Layout,” on page 274.](#)
- Updated Qp commands to support Roce, See [Section 29.10, on page 1897](#)
- Added Roce capabilities to Query_hca_cap, [Section 29.3.4.4, on page 1732.](#)
- Update Cqe format to support roce, See [Section 145, “64B Completion Queue Entry Format Layout,” on page 435.](#)
- Updated Cqe format to support Iro, See [Section 145, “64B Completion Queue Entry Format Layout,” on page 435.](#)

Revision 0.17 (June 2014)

- Updated networking chapter, cleared up text and added descriptions. See [Chapter 4, “Networking and Stateless Offloads” \(page 185\).](#)
- Updated introduction chapter. See [Chapter 1, “Introduction” \(page 162\).](#)
- **Added Vport counters commands.** See [Section 29.34, “Vport CountersMonitoring Commands,” on page 2176.](#)
- Added Q-counters commands. See [Section 29.35, “Q Counters Commands,” on page 2185.](#)
- Added *initializing* field. See [Table 14, “Initialization Segment,” on page 234.](#)
- Renamed translation_octword_size to translation_octword_actual_size in UMR WQE. See [Table 52, “UMR Control Segment Layout,” on page 323.](#)
- Renamed bsf_octword_size to bsf_octword_actual_size in UMR WQE. See [Table 52, “UMR Control Segment Layout,” on page 323.](#)
- Added new signature rules. See [Table 701, “Signature Rules Table,” on page 910.](#)
- Renamed *qp_counter_set* to *counter_set_id* in QP context. See [Table 19, “QP Context Structure Layout,” on page 258.](#)
- Added *counter_set_id* to DCT context. See [Table 25, “DCT Context Layout,” on page 279.](#)
- Updated NOP required segments. See [Table 106, “WQE Construction Summary,” on page 355.](#)
- **[internal] This PRM build is for first external release. No EATME updates.**

Revision 0.16 (May 2014)

- Updated wording of *rx_hash_fn*. See [Table 10, “TIR Context Fields,” on page 80.](#)
- Removed *rdma_pd* from RQ. See [Table 20, “RQ Context Fields,” on page 88.](#)
- Removed *counter_set* from RQ and SQ. See [Table 26, “SQ Context Fields,” on page 95](#) and [Table 20, “RQ Context Fields,” on page 88.](#)
- Clarified *sq_lst_sz* and *rq_lst_sz* are in entries. See [Table 10, “TIR Context Fields,” on page 80.](#)

- Removed *rx_hash_field_selector_ipv6_ex*. See [Table 9, “TIR Context Format,” on page 79](#) and [Section 4.1.3.2, “RX Hash Field Selection,” on page 82](#).
- Removed *MTU* from TIP. See [Table 10, “TIR Context Fields,” on page 80](#).
- Clarified *rq_lst_sz* that represents an indirection table must be an integral power of two. See [Table 10, “TIR Context Fields,” on page 80](#).
- TIP split into TIR (Transport Interface Receive) and TIS (Transport Interface Send). See the following sections:
 - [Section 4.1.3, “Transport Interface Receive \(TIR\),” on page 78](#)
 - [Section 4.1.4, “Transport Interface Send \(TIS\),” on page 84](#)
- Added *log_max_tir*, *log_max_tis* to QUERY_HCA_CAP. See [Section 22.1, “Networking Offload Capabilities,” on page 498](#).
- Added the relevant commands instead of TIS commands. See [Section 22.3, “TIR Commands,” on page 502](#) and [Section 22.4, “TIS Commands,” on page 507](#).
- Rename TIPN to TISN/TIRN in cqe format. See [Section 147, “Completion Queue Entry Fields,” on page 437](#).
- Added TIPN/TIRN and RQ Index/SQ Index to error cqe format, See [Table 159, “Completion with Error CQE Layout,” on page 455](#).
- Updated Network ports section. See [Chapter 3, “Network Ports” \(page 74\)](#).
- Renamed *vport_modify_permissions_allowed* to *esw_modify_allowed* in HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
- Updated Vport commands, See [Section 29.31, “Vport Commands,” on page 2154](#).
- Added *esw_vport* context. See [Section 9.4, “Esw_Vport Context - E-switch Virtual Port Context,” on page 121](#).
- Added *nic_vport* context. See [Section , “,” on page 287](#).

Revision 0.15 (March 2014)

- Removed *atm*, *log_max_atomic_size_qp* and *log_max_atomic_size_dc* from HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
- Added *atomic* to HCA_CAP. See [Table 1571, “HCA Capabilities Layout,” on page 1671](#).
- Added new *opc_mod* to support Atomic capabilities in QUERY_HCA_CAP. See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,” on page 1667](#).
- Added Atomic capabilities section. See [Section 29.3.4.3, “Atomic Capabilities,” on page 1728](#).
- Changed location of *free_ar* in QPC. See [Table 21, “Address Path Structure Layout,” on page 274](#).
- Increased *pkey_index* size in QPC to 16 bits. See [Table 21, “Address Path Structure Layout,” on page 274](#).

- Increased *pkey_index* size in DCT to 16 bits. See [Table 25, “DCT Context Layout,” on page 279](#).
- Added *pkey_table_size* to HCA_CAP, See [Section 1571, “HCA Capabilities Layout,” on page 1671](#).
- Moved *rss_hash_fn* and enlarged its size in TIP context. See [Table 9, “TIR Context Format,” on page 79](#). *rss_symmetric* and *rss_inner* field location was adjusted accordingly.
- Updated Flow Table Entry format and construction rules. See [Section 4.4.9.1, “Flow Table Entry,” on page 120](#).
- Updated format of the flow table entry in the flow table commands. See [Section 22.8.1, “SET_FLOW_TABLE_ENTRY - Set Flow Table Entry,” on page 528](#) and [Section 22.8.2, “QUERY_FLOW_TABLE_ENTRY - Query Flow Table Entry,” on page 530](#).
- Changes to *disp_type* field:
 - Added *disp_type* to CREATE_TIP command. See [Table 594, “CREATE_TIR Input Structure Field Descriptions,” on page 502](#).
 - Removed *disp_type* from MODIFY_TIP bitmask. See [Table 11, “MODIFY_TIR Bitmask,” on page 81](#).
 - Updated the field description in TIP context. See [Table 10, “TIR Context Fields,” on page 80](#).
-
- TIP Context Changes - see [Table 9, “TIR Context Format,” on page 79](#):
 - Renamed num_rqs and num_sqs to rq_lst_sz and sq_lst_sz
 - Removed rss_log_tbl_sz
 - Renamed rss_symmetric into rx_hash_symmetric
 - Renamed rss_inner into rx_hash_inner
 - Renamed rss_hash_fn into rx_hash_fn
 - Renamed rss_toeplitz_key into rx_hash_toeplitz_key
 - Removed rss_hash_type_ipv4, rss_hash_type_ipv6, rss_hash_type_ipv6_ex
 - Added new fields to control RSS hashing for IPv4,v6 and extensions: rx_hash_type_ipv4, rx_hash_type_ipv6, rx_hash_type_ipv6_ex
- Updated numbers, field names, rss table fields in MODIFY_TIP bitmask. See [Table 11, “MODIFY_TIR Bitmask,” on page 81](#).
- Added bit RX hash field selection section. See [Section 4.1.3.2, “RX Hash Field Selection,” on page 82](#).

Revision 0.14 (March 2014)

- Added TIP, SQ, RQ and RMP state description. See [Section 4.1.3.1, “TIR States Summary,”](#) on page 82, [Section 4.1.6.1, “SQ States Summary,”](#) on page 97, [Section 4.1.5.1, “RQ States Summary,”](#) on page 93 and [Section 4.1.7.1, “RMP States Summary,”](#) on page 100.
- Updated steering commands. Removed ADD_STEERING_RULE/MODIFY_STEERING_RULE/REMOVE_STEERING_RULE. Replaced them with SET_FLOW_TABLE_ENTRY/QUERY_FLOW_TABLE_ENTRY. The following sections are updated:
 - [Section 4.1.2, “Flow Table Entry,”](#) on page 78
 - [Section 4.3.9, “Flow Table,”](#) on page 120
 - [Section 22.2, “Steering Commands,”](#) on page 425
- Updated RSS section, RSS is handled through TIP. See [Section 4.3.6, “Receive Side Scaling \(RSS\),”](#) on page 194
- Updated TIP context, renamed transport to “st”, see [Section 4.1.3, “Transport Interface Receive \(TIR\),”](#) on page 78
- Updated SQ, renamed rlkey to rlky, updated required field list. See [Section 4.1.6, “Send Queue \(SQ\),”](#) on page 93
- Updated WQ, renamed wq_sig to wq_signature, renamed pg_ofst to page_offset See [Section 4.1.6, “Send Queue \(SQ\),”](#) on page 93
- Updated RQ, renamed rlkey to rlky, renamed inline scatter to cs, updated required field list. See [Section 4.1.5, “Receive Queue \(RQ\),”](#) on page 86
- Updated RMP required field list. See [Section 4.1.7, “Receive Memory Pool \(RMP\),”](#) on page 98
- Updated ODP description, See [Section 9.1.2, “On-demand Paging Support,”](#) on page 626.
- Added new opc_mod to support ODP capabilities in QUERY_HCA_CAP, See [Table 1568, “QUERY_HCA_CAP Input Structure Field Descriptions,”](#) on page 1667.
- Added ODP capabilities, See [Section 29.3.4.2, “ODP Capabilities,”](#) on page 1725.
- Updated MTT fields description, See [Table 366, “Memory Translation Table \(MTT\) Entry,”](#) on page 634.
- Added pg_access field to CREATE_MKEY command, See [Table 1797, “CREATE_MKEY Input Structure Layout,”](#) on page 1861.
- Added pg_access field to QUERY_MKEY command, See [Table 1807, “QUERY_MKEY Input Structure Layout,”](#) on page 1866.
- Added Page Fault Event, See [Table 175, “Event Type and Coding,”](#) on page 477.
- Added PAGE_FAULT_RESUME command, See [Section 29.6.4, “PAGE_FAULT_RESUME,”](#) on page 1869.
- Updated Sniffer commands, See [Section 21.25, “Sniffer Commands,”](#) on page 518.
- Added sniffer_physical_port to HCA_CAP, See [Table 1571, “HCA Capabilities Layout,”](#) on page 1671

- Added `block_lb_mc` to QP context, See [Table 19, “QP Context Structure Layout,”](#) on [page 258](#).
- Added `block_lb_mc` to HCA_CAP, See [Table 1571, “HCA Capabilities Layout,”](#) on [page 1671](#)
- Added `block_lb_mc` to CREATE_QP Required parameters See [Table 1885, “CREATE_QP Required Parameters,”](#) on [page 1902](#)
- Updated Network ports section, See [Chapter 3, “Network Ports”](#) (page 180).
- Updated subtypes of port State Change event. See [Table 204, “Port State Change Event Subtype,”](#) on [page 491](#).
- Added Vport context, See [Table 9.4, “Esw_Vport Context - E-switch Virtual Port Context,”](#) on [page 121](#).
- Added Vport Commands, See [Section 29.31, “Vport Commands,”](#) on [page 2154](#).
- Added `vport_modify_permissions_allowed` to HCA_CAP, See [Section 1571, “HCA Capabilities Layout,”](#) on [page 1671](#).
- Updated `vendor_error_syndrome` in Completion with Error to be 16 bits instead of 8 bits, See [Table 159, “Completion with Error CQE Layout,”](#) on [page 455](#)
- Added TIPN,SQ Index and RQ Index to CQE format, See [Table 147, “Completion Queue Entry Fields,”](#) on [page 437](#).

Notes for Sherin:

1. Table 1 conditional text legend - you should make it more readable by using the right colors, this is important, so please make the table with the colors so that people reading the PRM will know what the conditional text means

2. Must fix the conditional text. Too many conditions.

Revision 0.13 (January 2014)

- Revised [Chapter 4, “Networking and Stateless Offloads”](#) (page 185) to include new transport objects for Ethernet: TIP, RQ, SQ, RMP
- Revised [Chapter 22, “Ethernet Command Reference”](#) (page 498) to include new transport commands for Ethernet: TIP, RQ, SQ, RMP
- Renamed `sctr_grh_cqe` to `sctr_data_cqe` in [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,”](#) on [page 1667](#)
- Added new subtypes to port Event Change Event Subtype. See [Table 204, “Port State Change Event Subtype,”](#) on [page 491](#).

About this Manual

This Programmer's Reference Manual (PRM) describes the interface used by developers to develop Mellanox Adapters based solutions and to write a driver for Mellanox adapter devices. Specifically, it is intended for programming the devices corresponding to the PCI Device IDs listed in [Table 1](#).

Table 1 - PCI Device ID

Adapter Device	PCI Device ID (Decimal) - Physical Function	PCI Device ID (Decimal) - Virtual Function	PCI Device ID (Decimal) - Flash Recovery Mode
Connect-IB	4113	4114	511
ConnectX-4	4115	4116	521
ConnectX-4 Lx	4117	4118	523
ConnectX-5	4119	4120	525
ConnectX-5 Ex	4121	4122	
ConnectX-6	4123	4124	527
ConnectX-6 Dx	4125	4126	530
BlueField with crypto enabled	41680	41683	529
BlueField with crypto disabled	41681		
BlueField-2 with crypto enabled	41684		532
BlueField-2 with crypto disabled	41685		
ConnectX-4 Compatible Device Flash Recovery Mode (Future)			[TBD]
ConnectX-4 Compatible Device PF (Future)			4117, 4119, 4121, 4123, 4125, 4127, 4129, 4131
ConnectX-4 Compatible Device VF (Future)			4118, 4120, 4122, 4124, 4126, 4128, 4130, 4132
ConnectX-4 Compatible Device PF (Future)			4131, 4133, 4135, 4137, 4139, 4141, 4143

Table 1 - PCI Device ID

Adapter Device	PCI Device ID (Decimal) - Physical Function	PCI Device ID (Decimal) - Virtual Function	PCI Device ID (Decimal) - Flash Recovery Mode
ConnectX-4 Compatible Device VF (Future)			4132, 4134, 4136, 4138, 4140, 4142, 4144

Intended Audience

This manual is intended for software developers writing adapter device drivers.

The manual assumes familiarity with the InfiniBand and Ethernet architecture specification.

While generating documents always hide editor notes.

Document generation steps:

1. Open Shomron book only
2. Make sure the correct conditional text is on
 - Shomron PRM - Shomron Only, Editor Note, Internal, Dell, EMC, To-DO
 - Golan PRM - Golan Only, EMC, Redbook
3. Review revision history items
4. Track changes must be on (do not accept track changes)
5. Change bars must be on (do not delete changer bars)
6. Update book (cross-references, TOC etc.):
 - Update numbering of chapters manually before updating book
7. Search for keywords corresponding to text to be hidden in entire book to make sure nothing slipped out. The search is false-positive.
8. browse all figures to check if they are OK. This is because they are linked and conditional text is not applicable to fields in the picture, but rather to entire picture
9. change color and style of all conditional texts to “as is”
10. fix watermark if needed
11. update book again
12. generate the document (PDF or whatever needed)
13. browse the document to validate it is OK - sanity check.
14. Browse resulting PDF to validate (maker browse will not make it through pictures

Table 1 - Conditional Text Legend

Conditional Text	Color Scheme	Description
Golan Only	Dark Grey	Connect-IB only features
Shomron Only	Magenta	ConnectX-4 only features
Red book	Red	Confidential topics containing internal information. Can be disclosed under special NDA
EMC	Dark Blue	topics relevant to EMC only or exclusive to EMC. This text is to be exposed to documents sent to EMC only.
NotEMC	Muave/ Double Underline	topics which EMC should not see
HP	Salmon	for HP only
Golan-2 topics	Forest Green/ Underline	stuff relevant for Golan-2
Internal use	LinkBlue / Double Underline	(e.g. FW-related functionality, reserved fields). Must not be exposed in any external publication.
Editor notes	Magenta / Double Underline	(e.g. todo list)
Dell	Salmon / Double Underline	For Dell Only

Internal Notes to PRM Owner

Every feature is one of:

- 1 Stuff that is share-able with switch. For e.g. port configuration, temperature reading, power saving control, port counters, etc. We may consider even ACLs.
- 2 Stuff that is not share-able with the switch. E.g. memory registration and ODP, flow steering.
- 3 Stuff that falls in the gray area. E.g. in 2052 it may be shareable, eswitch config for HCA is sooo different today, all the command is defined already in Golan as command and just need to add few bits

It is up to the PRM owner to decide which category it is. And as a matter of fact it is a bit subjective.

Category 1 will be implemented using registers (ACCESS REG)

Category 2,3 will be implemented using command interface

Notes for the technical writer

Related Documentation

Table 2 - Reference Documents

Flash Programming for Mellanox Devices Application Note Document No. 2205AN	This application note describes how to manage firmware programming to a Flash memory device attached to a Mellanox Technologies adapter device
ConnectX® Programmer's Reference Manual (PRM) Document No. 2700PM	User guide describing the interface used by developers to write drivers between system software and Mellanox Technologies' ConnectX® devices
IEEE 1588-2008	IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.
InfiniBand Architecture Specification Release 1.2.1	IBTA specification
PCI Express® Base Specification	Industry Standard PCI Express 3.0 specification
Fibre Channel spec	
T10 DIF spec	
FICON spec	
EMC requirements document	
802.1 DCB extensions' documents	

TODO: update the table with right references to std. Sherin can you drive this?

TODO: add reference to VXLAN and NVGRE specs with the right version. Sherin?

TODO: update internal document numbers for new documents. Sherin can you drive this?

Documentation Conventions

Text and Figures

The document uses figures extensively to improve readability of the document. All figures are used for illustration purposes only and are not intended to replace written definitions. In case of conflict (mismatch) between figures and the text, the text definition and statements hold.

Terminology

The terms device, Host Channel Adapter device, HCA device, NIC, NIC device and adapter device are used interchangeably throughout this document to mean any of Mellanox's adapter devices. The terms Embedded CPU and BlueField family devices are used interchangeably.

Also, in the tables of this manual, if a field is said to be *set* without indicating a level of 0 or 1, it means that it holds the value 1. Similarly, if the field is said to be *clear* or *reset*, it means that it holds the value 0.

Byte Endianness

This manual uses Big Endian ordering of bytes. For fields greater than one byte in size, the most significant byte resides at offset 0 in memory. In the tables of this manual:

- Bits 31-24 reside in memory at byte offset 0
- Bits 23-16 reside in memory at byte offset 1
- Bits 15-8 reside in memory at byte offset 2
- Bits 7-0 reside in memory at byte offset 3

There are two exceptions to the Big Endian byte ordering scheme:

1. Atomics can be configured for processing in Little Endian byte ordering too per the Infini-Band specification.
2. PCI configuration headers are in Little Endian per the PCI specification.

Typographic Conventions

This manual uses the following typographic conventions:

- Command interface commands are capitalized, as in QUERY_ fields are denoted in bold italic font, as in *opcode_modifier*, or *l_key*
- Segments are denoted by italic font, as in the *Datagram* and *Bind* segments
- Unqualified logarithm base: Many instances of the mathematical function “log” appear throughout this document without qualifying the logarithm base--the reader should assume base 2 for all unless specified otherwise
- Table fields that are marked ‘reserved’ or left blank must be set to zero by *write* operations, and ignored by SW *read* operations

Part 1: Mellanox Adapters Operational Description

1 Introduction

This Programmer's Reference Manual (PRM) describes the interface used by developers to develop Mellanox Adapters based solutions and to write a driver for the supported adapter devices. The following Mellanox adapters are supported in this document:

- Connect-IB®
- ConnectX®-4
- ConnectX®-4 Lx
- ConnectX®-5
- ConnectX®-6
- ConnectX®-6 Dx
- [BlueField](#)
- [BlueField-2](#)

1.1 Major Features Summary

Table 3 - Summary of Features Per Device Adapter

Feature	Connect-IB	ConnectX-4	ConnectX-4 Lx	ConnectX-5
Virtual Protocol Interconnect (VPI)	InfiniBand only	InfiniBand and Ethernet	Ethernet Only	InfiniBand and Ethernet
InfiniBand Support and Maximum Speed	56Gb/s	100Gb/s	N/A	100Gb/s
Ethernet Support and Maximum Speed	N/A	100Gb/s	50Gb/s	100Gb/s
Network ports	1 or 2	1 or 2	1 or 2	1 or 2
PCIe	x16 Gen3	x16 Gen3	x8 Gen3	x16 Gen3 or x16 Gen4
Multi-Host	None	Supported (4)		Supported (4)
Performance (best theoretical) Message rate (MMsg/s) Bandwidth (Gb/s) Latency (us)	137 102 0.67	150 102 0.67		TBD
Transport Services	RC, UC, UD, XRC, DC	RC, UC, UD, XRC, DC, Eth	RC, UC, UD, XRC, DC, Eth	RC, UC, UD, XRC, DC, DCv2, Eth
RDMA over Converged Ethernet	None	RoCE, RRoCE/IP (pre std), RRoCE/UDP	RoCE, RRoCE/IP (pre std), RRoCE/UDP	RoCE, RRoCE/IP (pre std), RRoCE/UDP

Table 3 - Summary of Features Per Device Adapter

Feature	Connect-IB	ConnectX-4	ConnectX-4 Lx	ConnectX-5
Ethernet stateless offloads: TCP/UDP checksum, HDS, LSO, RSS	None	Supported	Supported	Supported
IPoIB, EoIB stateless offloads: TCP/UDP checksum, HDS, LSO, RSS	None	Supported	None	Supported
Eth, IPoIB, EoIB stateless offloads: TSS	IPoIB, EoIB-only	Supported	Ethernet only	Supported
Ethernet stateless offloads: LRO	None	Supported	Supported	Supported
QCN Congestion Control	None	Supported	Supported	Supported
Ethernet stateless offloads: Flow Steering	None	Supported	Supported	Supported
IPoIB, EoIB stateless offloads: Flow Steering	None	Supported	Supported	Supported
eSwitch ACLs Ingress (Rx)	None	L4, IPv4 and IPv6	L4, IPv4 and IPv6	L4, IPv4 and IPv6
eSwitch ACLs Egress (Sx)	None	L4, IPv4 and IPv6	L4, IPv4 and IPv6	L4, IPv4 and IPv6
Eth, IPoIB, EoIB stateless offloads: Tunneling (NVGRE, VXLAN)	None	Supported	Supported	Supported
Eth, IPoIB, EoIB stateless offloads: Tunneling (Geneve)	None	Eth only		Supported
Eth stateless offloads: MPLS	None	Supported	Supported	Supported
Eth Tunneling encap/decap offload (VXLAN, NVGRE)	None	Supported		Supported
Streaming Receive Queue (STRQ)	Supported	Supported		Supported
Erasur Coding offload	None	Supported	Supported	Supported
Signature Handover	Supported	Supported	None	Supported
Core Direct	Supported	Supported + vector calculation	Supported + vector calculation	Supported + vector calculation
Cache line align writes	None	Supported	Supported	Supported
Tag Matching and Rendezvous Offload	None	None	None	Supported
NVM Express over Fabrics (NVMe-oF)	None	None	None	Supported
Multi-Path QPs	None	None	None	Supported

Table 3 - Summary of Features Per Device Adapter

Feature	Connect-IB	ConnectX-4	ConnectX-4 Lx	ConnectX-5
Enhanced Multi Packet Send WQE	None	None	None	Supported
Fixed Buffer Size	None	None	None	Supported
Modify Packet Header (Header Rewrite)	None	None	None	Supported

1.2. Adapter Device Architecture Layers

The device implements a layered architecture, providing flexibility to use optimal infrastructure on each layer of the communication protocol, as shown in [Figure 2, “Architecture Layers,” on page 165](#). A few standard APIs are shown (as red lines) for completeness of the picture — their details are out of this document’s scope.

On the Transport Layer (Layer 4) the device implements InfiniBand transport services and Raw Ethernet transport services. These transport services can be independently assigned to either one of the 16M Flow Interfaces (Queue Pairs, or QPs – see below) implemented by the device, for each vHCA. Each QP can be associated with either network port.

The Network Layer (Layer 3) supports either IPv4 or IPv6 protocols and can be associated with either one of the Flow Interfaces (QPs).

The Link Layer (Layer 2) supports either the Ethernet or InfiniBand L2 protocol. The port configured to run the Ethernet layer, which can be accessed directly from the Ethernet NIC API, enables the operation of all Ethernet SW on the Ethernet port of the device. It can serve the InfiniBand network and the Transport Layer, enabling InfiniBand transport over the Ethernet / IP infrastructure, and providing a full set of InfiniBand features (channel IO, transport off-load, RDMA, and so on) to the client on the host. The Link Layer protocol of each port (Ethernet or InfiniBand) is determined either at boot time by the device, or dynamically according to the switch port type it is connected to.

The Physical Layer (Layer 1) supports either the XAUI (and other Ethernet Phy protocols) or the InfiniBand physical link, enabling a XAUI PHY, fiber and cabling infrastructure to be deployed in order to build both Ethernet and InfiniBand networks.

The device supports up to 16M independent IO channels, per vHCA. Each channel is individually configured to support a particular transport service and is associated with one of the device ports.

The architecture of the Transport, Link, and Physical Layers are illustrated below.

Figure 1: Basic InfiniBand Layers

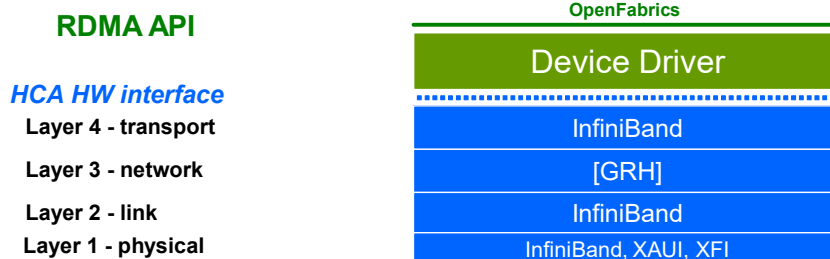
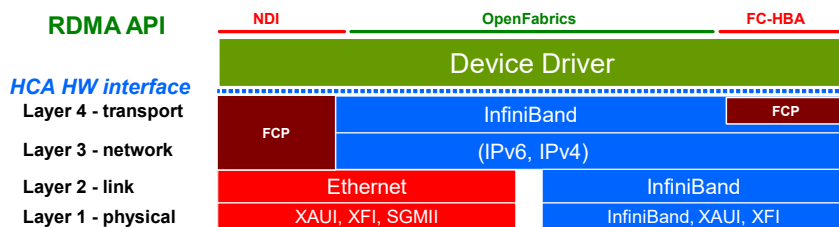


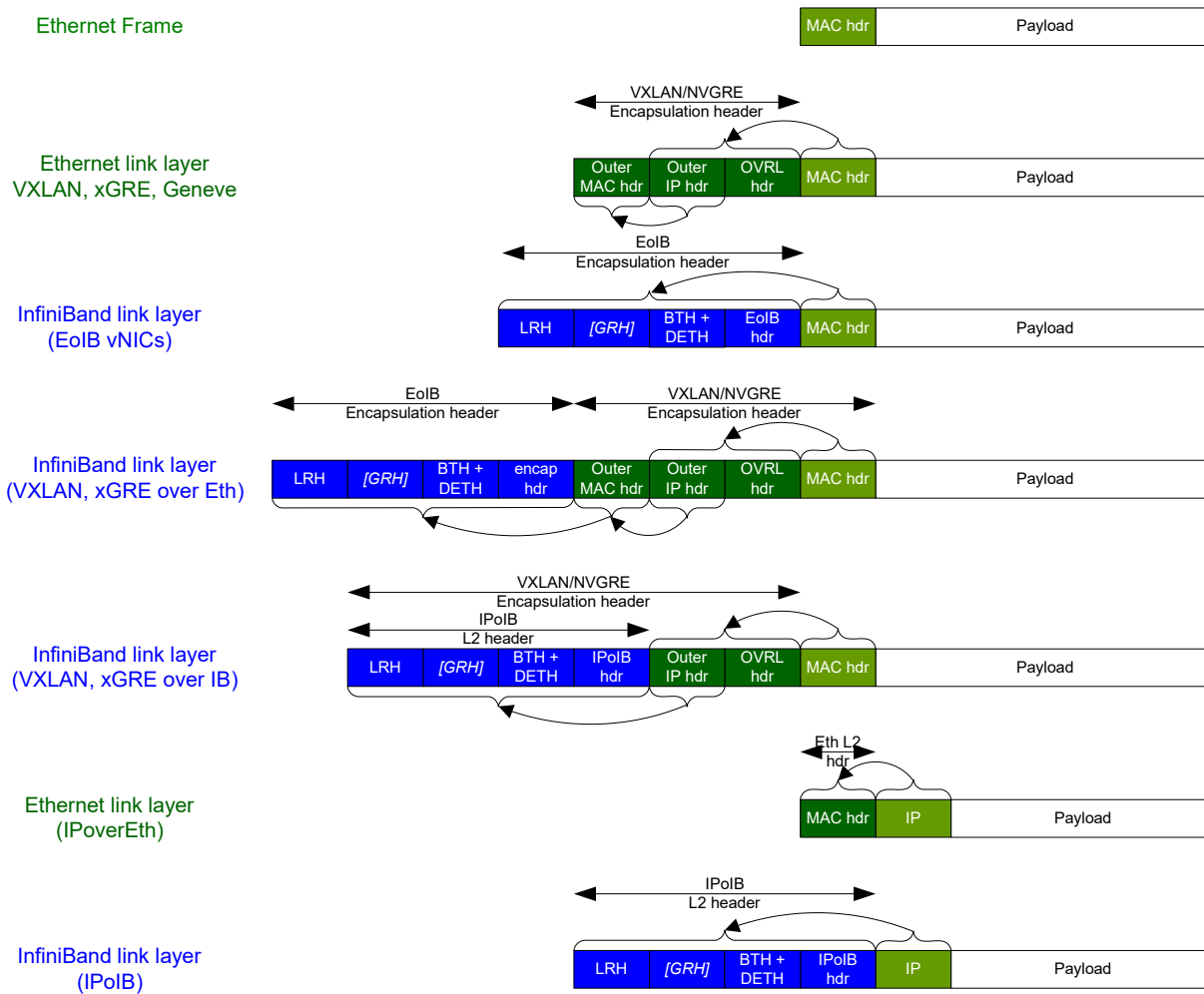
Figure 2: Architecture Layers



1.2.1. Wire Protocol

High-level encapsulation of wire protocol services is shown in [Figure 3](#).

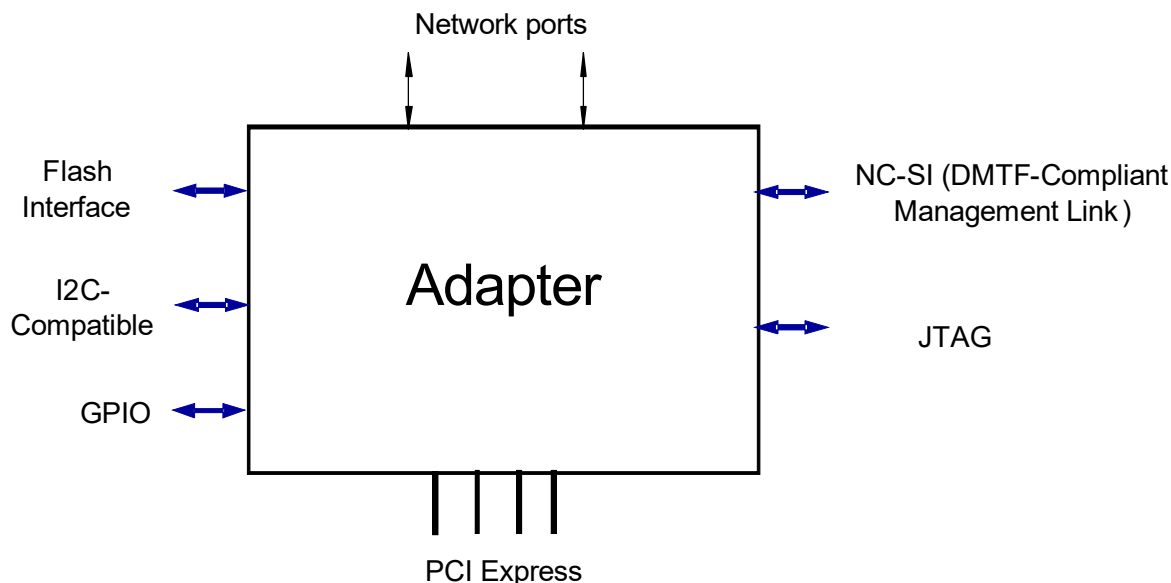
Figure 3: Wire Encapsulation



1.3 Hardware Interfaces

A high level illustration of the device is shown in [Figure 4](#).

Figure 4: System Interface



The device supports the following interfaces:

- Host Interfaces:
 - PCI Express – See [Section 1.3.1.1, “PCI Express,”](#) on page 168
 - NC-SI (DMTF-Compliant Management Link) – See [Section 1.3.1.2, “NC-SI \(DMTF-Compliant Management Link\),”](#) on page 168
 - I²C-Compatible – See [Section 1.3.1.3, “I2C-Compatible,”](#) on page 168
 - [Embedded CPU \(BF\) interface - Uses the PCI express interface. TBD](#)
- Network Interface – See [Section 1.3.2, “Network Interface,”](#) on page 168
- Flash Interface – This interface enables the HCA to perform the following:
 - Boot from a non-volatile memory (Flash) residing on this interface
 - Access the VPD data located on an NVMEM (Flash)
 - Access the expansion ROM for host boot over the network
- GPIO – the device can drive or read these pins. These pins can be used for various functions such as controlling LEDs on adapter cards, or as interrupt inputs to the HCA which can respond by issuing an event on the host interface. See relevant Adapters Datasheets for details.
- JTAG – IEEE-compatible JTAG access port

1.3.1 Host Interface - PCI Express

A PCI Express 3.0 (Gen3) interface is used as the primary host interface for controlling the adapter (HCA) device. For example, this interface is used for boot configuration and for posting WR. During operation, the HCA moves data between the host interface and the network interface. The HCA uses the host interface to access control structures (for example, context tables), report completions, etc. Consumer application controls HCA operation through the host interface (for example, by ringing DoorBells).

The host interface can be divided into up to 4 independent PCIe interfaces. For more details on this Multi-Host capability.

1.3.1.1 PCI Express

The PCI Express interface is fully compatible with PCI-Express 3.0 specifications, implementing 16 lanes (x16) and delivering host memory access bandwidth of 16GBytes/sec in each direction. Various HCA products support lower number of lanes and lower signaling rate.

The software and management part of the PCI interface is discussed in [Chapter 5, “PCI Interface”](#) on page 226.

1.3.1.2 NC-SI (DMTF-Compliant Management Link)

NC-SI interface is a DMTF-compliant interface used to connect the device to an external Baseboard Management Controller (BMC). The device supports NC-SI version 2.0. Please contact Mellanox Technologies for more details.

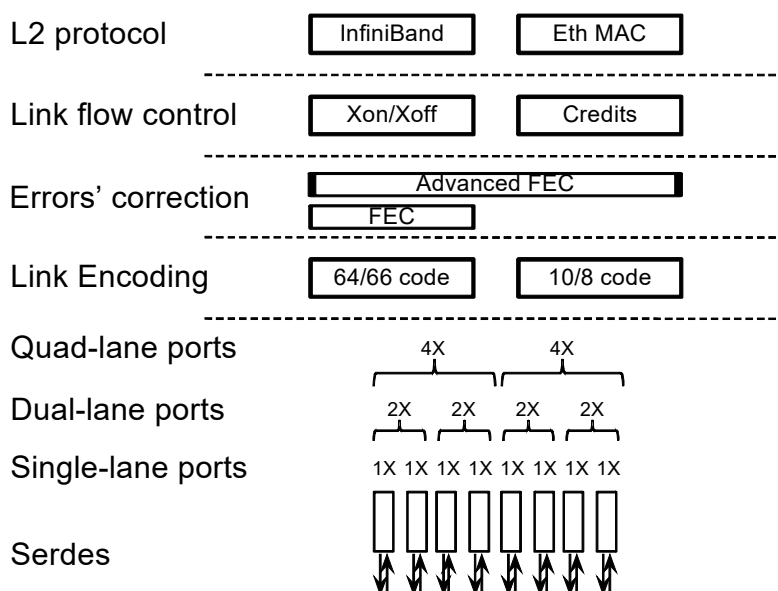
1.3.1.3 I²C-Compatible

The I²C-compatible interface is designed to enable out-of-band access to the device, chassis management (IBML/SMBUS), and is used for HW debug. This interface is also designed to enable Mellanox debug tools to access the device, even in the event of other interfaces being down. The adapter device can be either a master or a slave on this interface (slave address is 0x48), and can support two different I²C-compatible interfaces for redundancy/fault tolerance.

1.3.2 Network Interface

The network interface is comprised of 4 or 8 high-speed SERDES lanes running at up to 25Gb/s each. Utilizing these SERDES lanes, the HCA provides one or two 4X network ports. Each network port can use one, two or four lanes. Single-lane port configuration can use any lane. Multi-lane port configuration is supported on top of *pairs* (2X) or *quads* (4X) of SERDES lanes. Depending on the number of port lanes and lane speed configuration, each network port can run at 10, 25, 40, 50, 56 or 100Gb/s.

Figure 5: Ports' Configuration



Each network port can be independently configured to use either single, dual or quad lane configuration. Signaling rate of each port can be configured independently.

On top of each logical port, HW implements multiple layers of link reliability architecture, enabling the adjustment of physical characteristics of signal transmission to the media and operational conditions (for example link encoding, FEC, LLR etc). **This is discussed in “Unbreakable Link Architecture” document.**

Next layer is link-level flow control. The HCA can support credit-based flow control.

2 Operational Overview

This chapter provides an overview of the device operation, describing the data structures and control mechanisms of the device in general.

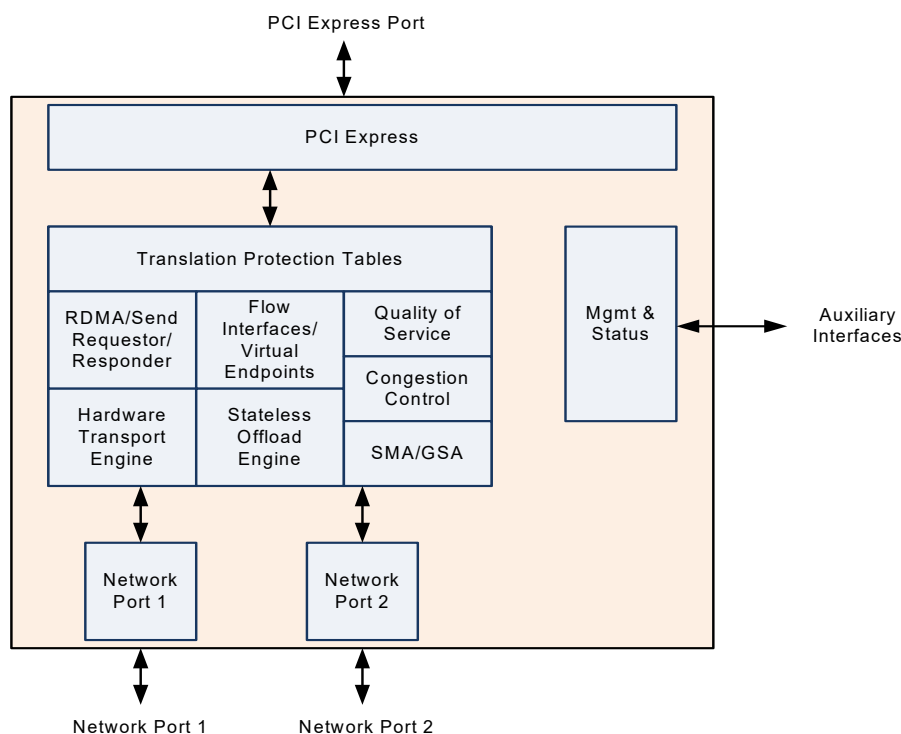
The device operation is described in the following sections:

- [Section 2.1, “Operation Interfaces,” on page 170](#)
- [Section 2.2, “Software Interface,” on page 171](#)
- [Section 2.3, “HCA Configuration,” on page 171](#)
- [Section 2.4, “HCA Operation,” on page 171](#)
- [Section 2.5, “Virtualization,” on page 179](#)
- [Section 2.6, “Quality of Service,” on page 179](#)

2.1 Operation Interfaces

The device has a single main host interface (PCI Express) and multiple auxiliary host interfaces (Fast Management Link, DMTF-compliant management interface, I2C) to the host agents such as BMC. Each of these interfaces can be used to control HCA operation, configure the device, and pass data to and from the network. For example, the device can be booted from either a PCI Express or an auxiliary interface.

Figure 6: Operation Interfaces



Network access from an auxiliary host interface is implemented through Flow Interfaces (FIs) associated with UD InfiniBand transport services. Network traffic associated with these FIs is passed to the respective auxiliary interface.

The HCA determines its boot interface through the boot record which it reads from an attached flash device.

2.2 Software Interface

The device can be accessed by system and application software in the following ways:

1. Through the PCI configuration interface. This access type is used during the PCI Express fabric enumeration to configure the device.
2. Through the HCA Configuration Registers that are accessed through memory BAR0 (see [Chapter 5, “PCI Interface” on page 226](#)). The software initializes the device and configures basic device capabilities at boot using this interface.
3. Through user-level access to the HCA hardware (kernel bypass), using the User Access Region (UAR), as described in [Section 8.2, “User Access Region \(UAR\),” on page 249](#).

2.3 HCA Configuration

At initialization, the device is configured through an HCA command queue (see [Section 8.28.1, “HCA Command Queue,” on page 613](#)) and HCA registers. At this stage, the device capabilities (amount of resources to be supported), port capabilities and HCA resources are configured.

The HCA registers (UAR/DoorBells and initialization segment) are mapped to physical memory space. The mapping is provided through a standard PCI BAR/LIMIT mechanism. For more details, please refer to [“PCI Interface” on page 226](#).

The command interface is used for general HCA device configuration. For more details, please refer to [Section 8.28, “Command Interface,” on page 613](#).

2.4 HCA Operation

After the HCA is initialized and opened, FIs comprised of Work Queue (WQ) pairs can be established and data transfer can start. the host software supports send and receive data transfers through Work Requests (WRs) posted to Work Queues (WQs). Each WQ contains a Send Work Queue (SQ) for posted send requests, and a Receive Work Queue (RQ) for posted receive requests (see [Chapter 8, “Software Interface” on page 248](#)). In addition to send requests, WRs posted to an SQ may also initiate RDMA read, RDMA write and Atomic operations, as well as memory translation/protection configuration operations. Each posted WR that requires data movement generates a single IB message which consists of one or more IB packets.

The WR is posted as a Work Queue Entry (WQE) to an SQ/RQ. These WQEs can either cause data to be transmitted (Send or RDMA Write request) or received (RDMA Read request). WQEs are essentially descriptors that control the source and destination of data movement.

WQEs that transmit data from the HCA local memory can include a “gather” list. This “gather” list points to a set of memory buffers from which to assemble the outgoing message or contain immediate data to be sent. Similarly a WQE requesting an RDMA Read includes a “scatter” list, specifying the location where incoming data will be placed. When the RDMA read response

message is received, the data will be “scattered” to a local memory based on the WQE. The RQ also has a “scatter” list included in posted WQEs to indicate memory buffers to be used for placing the received send payloads.

Data transfer is initiated by posting a descriptor to an SQ and ringing the respective DoorBell. Once the DoorBell has been rung on an SQ, the HCA initiates execution of the send descriptor, reading the descriptor and executing it according to the descriptor’s opcode - data transfer, memory registration and so on.

After a descriptor is executed and the message is completed, the HCA will post an entry to the corresponding Completion Queue (CQ). Multiple WQs can report their completions to the same CQ; the CQ number to report the completions is specified for each WQ at its initialization and is held by the HCA in its QP Context (QPC) structure. The HCA stores a WQE identifier in the CQ Entry (CQE) in order for software to figure out which descriptor is completed and on which QP, enabling efficient multiplexing of multiple QPs on the same CQ as well as implementing ‘completion coalescing’ in software.

2.4.1 Send Operation Overview

For a send operation, host software posts a WR on an SQ. HCA hardware places this request on one of the schedule queues. The schedule queues implement Quality of Service (QoS) levels for scheduling messages for execution.

When initialized, each QP is assigned a fixed number of SQs and is mapped to one of the QoS levels. HCA hardware arbitrates between the QoS levels to decide which QP is to be scheduled for execution.

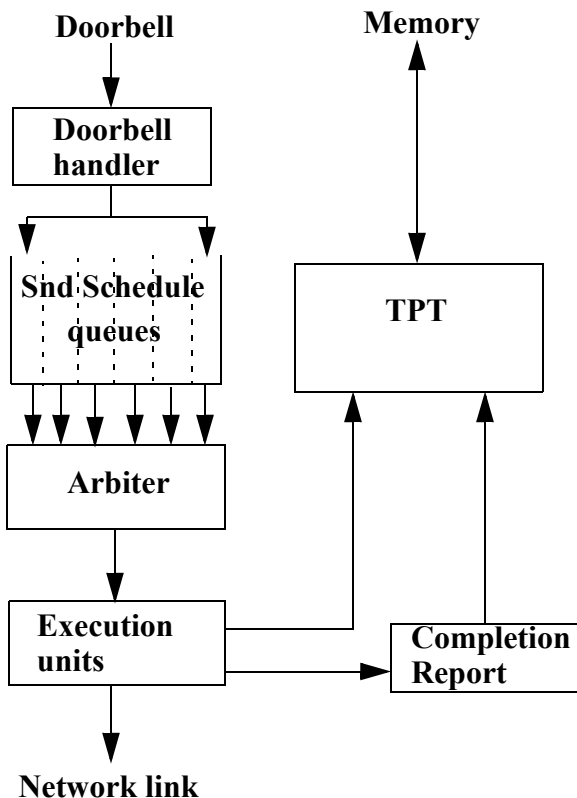
The association between SchQ and QP is done at the QP initialization stage. When a DoorBell is rung on an SQ, HCA hardware places the SQ’s QP on the tail of its schedule queue to await later scheduling. Whenever HW resources are available, the QoS mechanism(See [Chapter 21, “Quality of Service” on page 237](#)) selects the next schedule queue to be processed. Once a schedule Queue is selected, the head QP on that schedule queue will then be selected for processing.

Hooks are provided for the programmer to control HCA or network resource allocation per schedule queue (QoS provisioning). QP Context maintenance and all address translation and memory access permission checks are implemented in hardware while maintaining full wire speed of the host interface and/or network links.

Any operation which requests access to memory passed first through the TPT, which is responsible to perform all the required protection checks and address translation.

For unreliable transport service, the send completion is reported as soon as the message is transmitted. For reliable transport services, the send completion is reported after the last packet of the message is ACK’ed by the remote node. If a transport error is encountered, the HCA will attempt to deliver the message (for example re-send message or part of it, execute automatic path fail over and so on) through all mechanisms provided by the transport specification and HCA configuration. If the message still cannot be delivered after all remedy options are exhausted, the HCA will report an error to the CQ and transfer the QP to the appropriate error state.

[Figure 7](#) illustrates the send operation flow.

Figure 7: Send Operation Flow

2.4.2 Receive Operation Overview

Once a message is received by the HCA from the network, transport checks are performed and the request is passed to the receive processing engines for execution. Sends and RDMA-write requests are executed in the order they are received on the wire within the same QP. RDMA-read requests are scheduled for execution through a set of schedule queues dedicated to handling RDMA-read requests. Mapping of a particular RQ to a schedule queue is set by software at QP initialization. HCA receive engines will arbitrate between all requests received (RDMA-read responses and ACKs) and schedule them for execution.

The HCA contains integrated input buffers on the network port to temporarily store ingress traffic. The buffers are tuned to serve as a shock absorber between receive message rate and availability of HCA resources to process incoming messages. **Once the buffers are full, it will cause back pressure on the network. After a programmable back pressure time-out is exceeded, the HCA will drop a programmable number of packets. Both time-out and number of messages dropped can be configured such that the HCA will either never assert back pressure or never drop messages. Tuning of these parameters is left for system design discretion.**

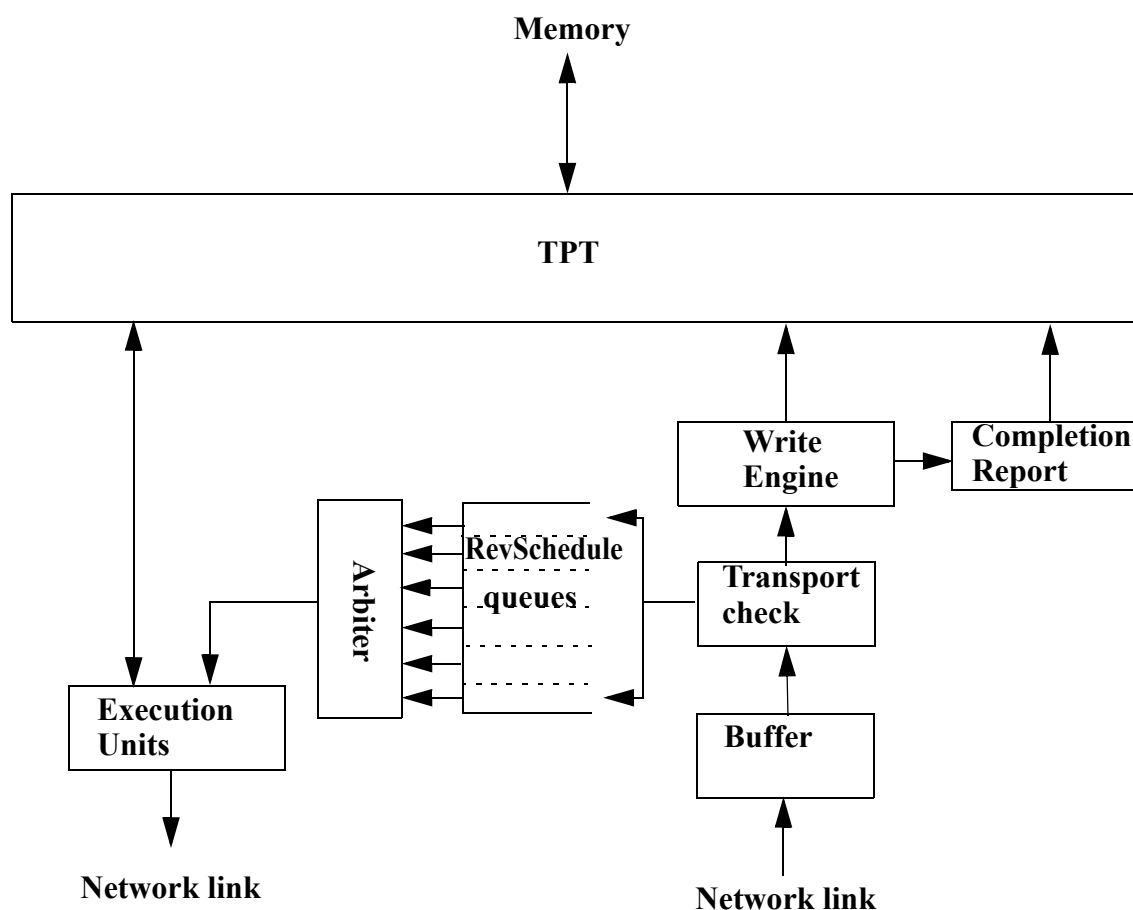
The HCA limits the number of RDMA-read requests that can be received on any QP opened to the maximum which is defined at QP creation. Each QP is allocated enough memory resources at its initialization to hold all outstanding RDMA-read requests.

The HCA implements network operations using posted write semantics. The ACK for RDMA-writes or sends is scheduled after transport checks are completed, memory access rights are validated, and the write is posted to the target's host interface. Thus, transport-level ACK does NOT guarantee that data arrived to its ultimate destination (receive buffer). Software must implement a higher-level acknowledgment to assure that the ultimate consumer of the data (the host application and not memory hardware) received the data.

On the target-side, the HCA reports completions to the CQ configured for the RQ for each send or RDMA-write with Immediate. RDMA-read requests are invisible to the target driver and it has no knowledge of their processing.

Receive operation flow is shown in [Figure 8](#).

Figure 8: Receive Operation Flow



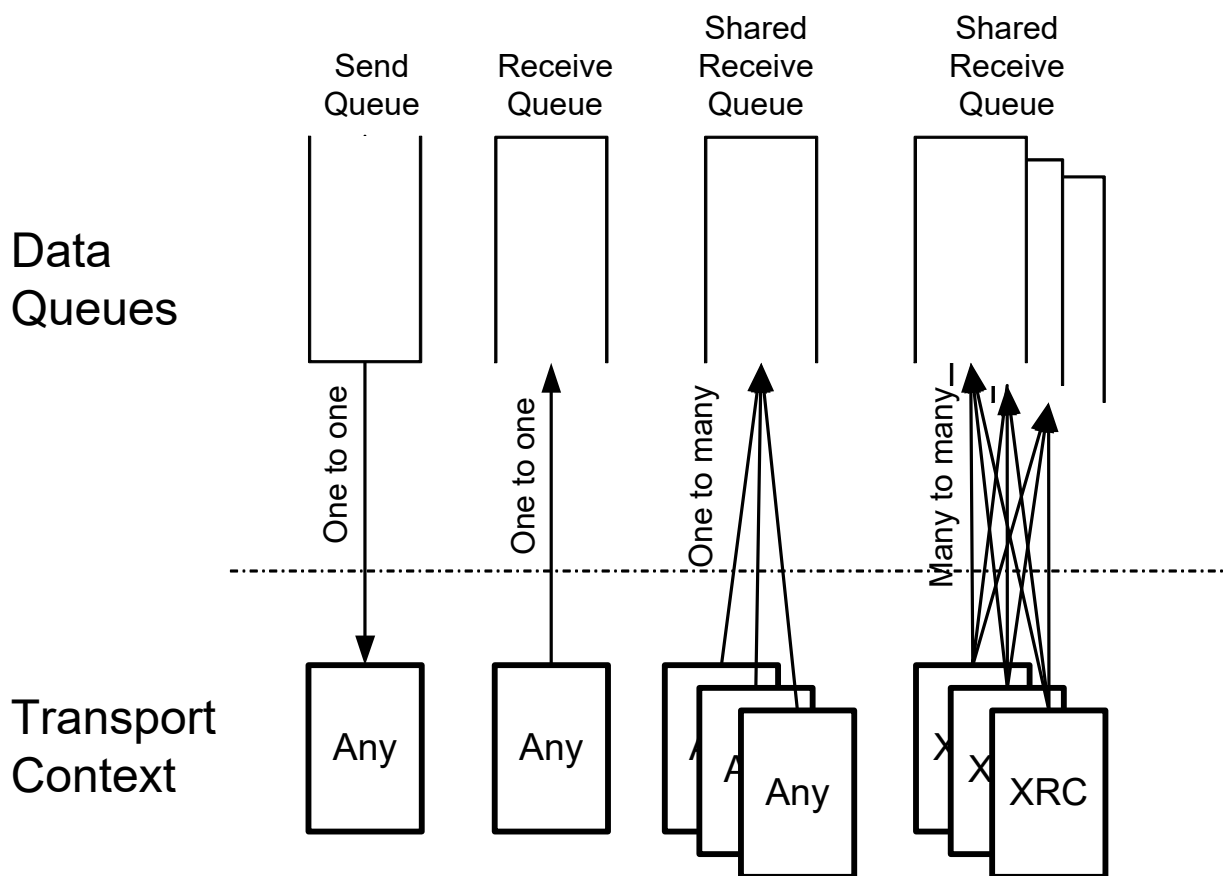
2.4.3 Queue Pairs

Network and I/O services are provided to the consumers via Flow Interfaces. The device supports a single type of FIs called Queue Pairs (QPs), and the two terms are used interchangeably.

The device implements 512 virtual HCAs and each one can have up to 16M FIs. Each FI is associated with one of the network ports of the device.

An FI is comprised of a pair of WQs associated with one or more Transport Contexts (TCs) which implement the transport services. Send WRs are posted to an SQ. These requests initiate data transfer from the consumer or execute local control operations. Receive WRs are posted to a RQ to receive data from the network. See [Figure 9](#).

Figure 9: Relation Between Transport Context and Receive Queue



QPC is a control data structure describing QP properties. This structure is generated by the HCA at QP creation and is used by the HCA to control I/O operations associated with this QP. Once a QPC is generated, it is maintained by the HCA in the Interconnect Context Memory (ICM). It can be retrieved by curious SW to see what is going on, but this is left for advanced users and is not recommended to try at home. TC information, relations between SQ, RQ and TC, memory protection information and so on are maintained in the QPC. For more details, please refer to [Section 8.3, “Queue Pairs,”](#) on page 254.

2.4.3.1 Special QPs

The HCA can forward packets arriving on special QPs, QP0 and QP1, to the device driver. For more details, please refer to [Section 8.3.4, “Special QPs,”](#) on page 255.

2.4.4 Flow Interface Isolation and Protection

Each FI is a consumer interface to an independent I/O channel. The channel properties (transport service, amount of I/O resources allowed to be consumed, execution priority and characteristics) are set by the privileged entity (driver), and the device enforces cross-channel protection and isolation at run time. Thus, hardware assures that each FI does not consume more I/O resources than its allocated amount. Hardware also enforces protection and isolation; for example, error or slow-down in operation on one FI will have no effect on other FIs.

2.4.4.1 Raw Network Interface – Mellanox Transport

The HCA can send raw data to the network by configuring QPs as “Mellanox (MLX) transport”. Packets of MLX transport QPs are generated without headers, and are sent to the wire without performing checks and no acknowledgment is expected.- in case of uncontrolled injection due to SW bugs is not enough, and can even generate packets with bogus CRC - something that SW cannot do on its own. See Section 8.3.4.3, “Mellanox Transport,” on page 256.

2.4.5 Work Queues

Mainstream operations (send/receive data, bind and so on) are posted to WQs by the application SW. The Work Queue is accessible directly by the application program to post Work Requests to the HCA. Each Work Queue Element (WQE) represents a single Work Request. The Work Requests posted to WQs are executed by the HCA in the order that they are posted. Work Request is posted to the HCA by writing a list of one or more WQEs to the WQ and ringing the DoorBell which notifies the HCA that the request has been posted. For more details, please refer to [Section 8.9, “Work Queues,” on page 298](#).

2.4.5.1 Shared Receive Queue Overview

Shared Receive Queues (SRQs) enable associating multiple RQs with a single pool of receive buffers, allowing better memory utilization and management by pre-posting a receive buffer to a single pool that can serve multiple receive transports.

When a QP is initialized, it can be associated with an SRQ. Association with an SRQ does not impact any send or responder operations that do not consume receive WQEs (RDMA-read, Atomic or RDMA-write without immediate). Upon reception of a WQE-consuming request on the RQ associated with an SRQ, the WQE is dequeued from the shared list (SRQ) and this WQE controls the operation of the received request.

SRQ-attached Reliable Connection (RC) QPs report invalid credits when generating Ack messages. This prevents remote requesters from getting into the limited state. However, if messages arrive while the SRQ has no receive descriptors available, an RNR Nak is sent on the QP (RC) or the message is dropped (UD). Correct provisioning of buffer pre-posting yields the best overall system performance, avoiding the RNR Nak flow (RC QPs) or dropping of packets (UD QPs). For more details, please refer to [Section 8.10, “Shared Receive Queue \(SRQ\),” on page 365](#).

2.4.6 Completion Queues and Completion Events

The HCA supports up to 16M CQs and enables reporting the completion of send or receive requests to different CQs. Multiple WQs can report their completions to the same CQ, maintaining many-to-one relations between WQs and CQs.

Send completion reports can be optional if a QP is configured as ‘un-signalled’. In this case, completion reporting is controlled by a bit in the send WQE.

Events (and possibly a subsequent interrupt) can be registered for a CQ and are generated when a Completion Queue Element (CQE) is placed on that queue.

For more details, please refer to [Section 8.19, “Completion Queues,” on page 432](#).

2.4.7 Event Queues

Events generated by hardware are posted to Event Queues (EQs). The EQ is a memory-resident circular buffer used by hardware to write event cause information for consumption by the host software. HCA has multiple sources that can generate events (completion events, asynchronous events/errors). Once an event is generated internally, it can be reported to the host software via the Event Queue mechanism. When event reporting is enabled, event cause information is written by hardware to the EQ when the event occurs. If EQ is armed (that is, at least one event has been written to EQ), HW will subsequently generate an interrupt on the device interface (send an MSI-X message or assert the pin) as configured in the EQ.

Each one of the HCA EQs can be associated with a different event handler on the host. The HCA supports multiple EQs designed to enable de-multiplexing of events to different consumers on the host. For example, completion reports can be reported to different EQs, based on the CQ that reports the event. Each CQ can be configured to report its events to a particular EQ; multiple CQs can report events to the same EQ. A completion event generated on the CQ may cause an event to be generated on the EQ that a certain CQ is mapped to.

In a virtual environment, EQs can be exported to the guest Virtual Machine (VM), and the kernel driver in the guest VM will control the EQ. HCA HW enforces protection and isolation between EQs.

For more details, please refer to [Section 8.20, “Events and Interrupts,” on page 472](#).

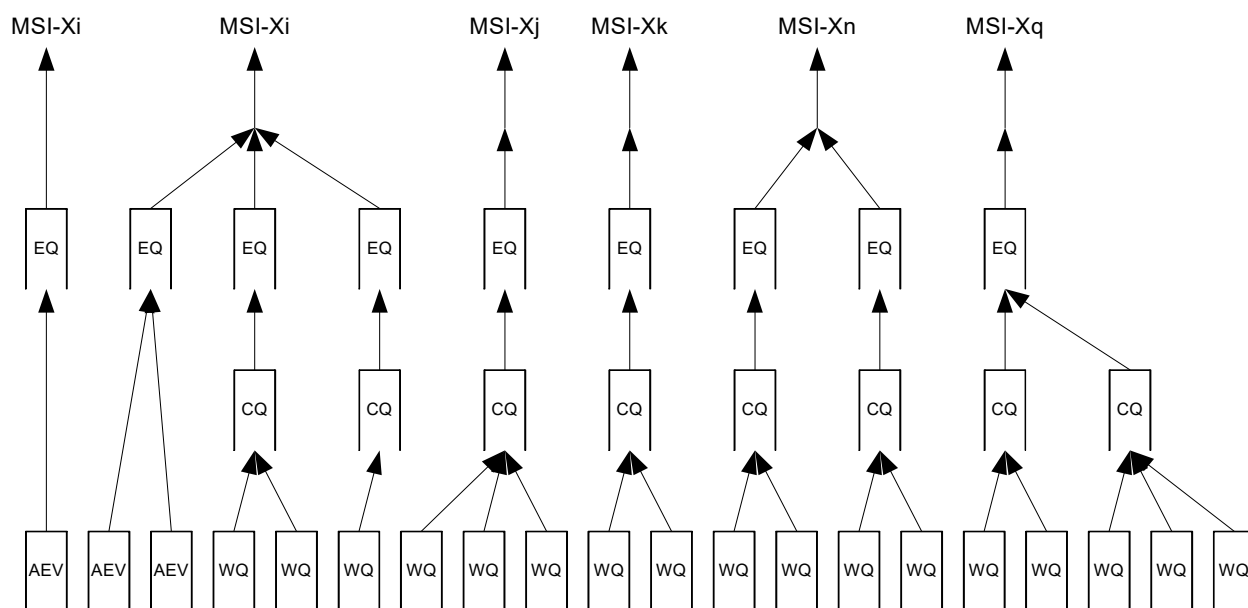
2.4.7.1 Interrupts

The HCA supports multiple means of generating interrupts - asserting a pin on its physical interface, emulating interrupt pin assertion on the host link (PCI) or generating Message Signaled Interrupts (MSI/MSI-X), enabling software to de-multiplex interrupts to different host consumers.

Each EQ can be configured to generate an interrupt when an EQE is posted to that EQ. Multiple EQs can be mapped to the same interrupt vector (MSI-X) maintaining many-to-one relations between EQs and interrupts.

The relations between WQs, CQs, EQs and different interrupt messages (MSIX vectors) is shown in [Figure 10](#).

Figure 10: WQs, CQs, EQs and Interrupts Relations



Asynchronous events - like link state change or various errors - can also cause an event to be posted and an interrupt to be asserted. Each asynchronous event type can be mapped to a specific EQ and optionally generate an interrupt. Hardware does not prevent mapping synchronous and asynchronous events to the same EQ. The user should use common sense while configuring a device and use distinct EQs for asynchronous events.

2.4.8 HCA Memory Resources

Memory structures controlling HCA operation reside in the host memory allocated for the HCA by the host software. This memory - called Interconnect Context Memory (ICM) - is allocated by SW upon HCA request (in page-size chunks) and passed to HCA ownership. From that point on, SW should not access this memory until it is explicitly passed back to SW ownership by the HCA. Once given to the HCA, the HCA manages and uses this memory at its own discretion. For details see [“Interconnect Context Memory” on page 239](#).

The amount of ICM memory used depends on the amount of HCA resources that will be used as in the following:

- Address Translation Tables - control structures in ICM memory allocated by the HCA for every memory key
- HCA Control Objects (Contexts) - control HCA operation and are managed by the HCA in its ICM
- User Access Region - a memory region that can be mapped directly to an un-trusted application by the OS and can be addressed by non-privileged code

For more details, please refer to [“Memory Resources and Utilization” on page 239](#).

2.5 Virtualization

Mellanox HCAs support Virtualization, implementing full isolation and protection between flow interfaces and exposing management interfaces to provision cross-channel isolation.

The HCA can expose multiple physical and virtual functions on the PCI bus, enabling different functions to be mapped to guest Virtual Machine (VM) domains and independent management (that is, shut-down) of each individual function without causing any harm to any of the other ones.

On the network side the device enables virtual switch offload.

For more details, see [Chapter 14, “Virtualization” on page 838](#). [At some point host stack \(ULPs\) overview will be added here.](#)

2.6 Quality of Service

The device supports end-node Quality of Service implementing a 3-stage arbitration between different FIs with pending network requests. The device is capable of enforcing guaranteed bandwidth or guaranteed latency to selected FIs, while providing “best effort” service to others.

3 Network Ports

3.1 Port Model Overview

The virtualization port model is described in details in [Section 14, “Virtualization,” on page 838](#).

3.2 Physical Port

3.2.1 Overview

The device supports Unbreakable-Link™ technology to enable optimal performance of its links under different channel constraints.

The ports’ personalities can be configured as ini parameters on a per adapter/LOM basis. In addition, NVRAM configuration can be used to configure the runtime selection of protocols on VPI cards.

Editor Note - Golan OverView:

The Connect-IB® is a dual port device that supports InfiniBand protocol with different speeds up to EDR. Connect-IB supports all base standard features. In addition, Connect-IB supports Unbreakable-Link technology to enable optimal performance of its links under different channel constraints.

The ports personalities can be configured as ini parameters on a per adapter/LOM basis. In addition, NVRAM configuration can be used to configure the runtime selection of protocols on VPI cards.

[Internal] R- Base R 64/66 NRZ , X 8/10 relevant for 1G,10G.

C - copper , K - backplane , S - short range optics, L - Long range optics , E - very long optics

3.2.2 Port Speed Capabilities

<Editor Note> Connect-IB has just second table

ConnectX-4/Connect-IB™ device supports the following feeds and speeds:

Table 4 - Summary of ConnectX-4 Ethernet Port Speeds

Mnemonic	Speed
100GBase-LR4/ER4	100G
100GBase-SR4	100G
100GBase-CR4	100G
100GBase-KR4	100G
56GBase-*R4	56G
50GBase-*R2	50G

Table 4 - Summary of ConnectX-4 Ethernet Port Speeds

Mnemonic	Speed
40GBase-ER4/LR4	40G
40GBase-SR4	40G
40GBase-CR4	40G
40GBase-KR4	40G
40GBase-*R2	40G
25GBase-*R	25G
20GBase-*R	20G
20GBase-CR2	20G
20GBase-KR2	20G
10GBase-CX4	10G
10GBase-KX4	10G
10GBase-LR/ER	10G
10GBase-SR	10G
10GBase-KR	10G
10GBase-CR	10G
RXAUI	10G
1000Base-KX	1G
1000Base-CX	1G
SGMII	1G

Table 5 - Summary of ConnectX-4/Connect-IB™ InfiniBand port Modes

Mnemonic	Speed ^a
EDR	100G
FDR	56G
FDR10	40G
QDR	40G
DDR	20G
SDR	10G

a. Gross speed, calculated for x4 lanes configuration

3.2.3 Port Numbering

3.2.4 Congestion Control

The device supports, in hardware, multiple end to end congestion control protocols. For InfiniBand, it fully supports Annex 10 of the InfiniBand spec. For general traffic on top of Ethernet network, it supports QCN (IEEE standard 802.1Qau). For RoCEv2, the device supports RoCEv2 Congestion Management protocol according to the definition in Annex 17 of the InfiniBand spec (RCM, section A17.9.3).

The hardware supports activating either one of these protocols at any given time for each port.

The device hardware allows configuring protocol specific parameters for each of the congestion control protocols in Ethernet mode. The configuration of these parameters is done using the commands `QUERY_CONG_PARAMS` and `MODIFY_CONG_PARAMS` (See [Section 29.41, “Congestion Control Commands,” on page 2228](#)). These commands also allow the user to configure what protocols are enabled for each port. Note that the InfiniBand congestion control parameters will be configured by the subnet manager, through MAD messages, as defined in the InfiniBand spec.

The adapters congestion control hardware also allows the user to query statistics and monitor the congestion situation using the `QUERY_CONG_STATISTICS` command.

Note that querying congestion parameters or statistics is allowed only if `HCA_CAP.cc_query_allowed == 1`, and modifying congestion parameters is allowed only if `HCA_CAP.cc_modify_allowed == 1`. (See [Table 1571, “HCA Capabilities Layout,” on page 1671](#))

3.2.4.1 QCN Congestion Control

The QCN IEEE standard (802.1Qau) provides congestion control for long-lived flows in limited BW-delay product Ethernet networks. It is part of the IEEE Data Center Bridging (DCB) protocol suite, which also includes ETS, PFC, and DCBX. QCN is conducted at L2, and is targeted for HW implementations. QCN applies to all Ethernet packets and transports, and both the host and switch behavior is detailed in the standard.

The device implements QCN in HW, in a transport agnostic manner. SW is not involved in the data path. However, drivers do control which Ethernet packets are transmitted through which QPs, that constitutes congested controlled flows.

The parameters provided for controlling QCN are the same as defined in the QCN standard. The hardware provides all counters required by the standard as well as additional counters.

3.2.4.2 RoCEv2 Congestion Control

The adapter hardware fully implements the RoCEv2 Congestion Management defined as part of the Annex 17 of the InfiniBand spec.

The hardware uses a mechanism used by DCTCP to determine the congestion state of the network based on the CNP received. This estimation is used to drive a mechanism which is based on the QCN protocol to reduce and increase the rate. The round trip time estimation and averaging factor for DCTCP congestion level estimation, as well as the QCN protocol parameters are configurable per port.

The adapter hardware also allows configuring the DSCP and priority values in the CNP packets, as well as rate limitation of the number of CNPs sent from each port.

3.2.5 Port States

3.2.6 Port Up/Down Flow

<TBD> how link up/down, which configurations needed,

3.2.7 Flow Control

3.2.8 Buffer Management

3.2.9 Cable Management

3.2.10 Port Control Registers

The adapters' ports parameters can be configured/queried by a set of port control registers (See [“Network Ports Registers”](#) on page 1140).

3.2.11 Mellanox Unbreakable Link Technology

<TBD> check what of the bellow supported in Golan/Shomron

ConnectX-4/Connect-IB™ supports the Unbreakable Link Technology. This technology is supported on links that connect devices that support and enable the technology. Unbreakable Link Technology provides optimal performance of the link tuned to the actual channel quality and media type.

Unbreakable Link Technology comprises:

- Link speed tuning during negotiation
- On the fly link speed change
- Persistent link monitoring
- Low Latency Reed Solomon FEC - optimized FEC algorithm
- Link Level Retransmission - LLR and PLR ([TBD: what is PLR](#))
- Adaptive LLR and PLR
- Width tuning (x1, x2, x4)
- Mellanox Link Protocol Negotiation (MLPN)

3.2.12 FEC/RS-FEC

3.2.13 LLR/PLR

3.2.14 Communication Block

3.2.15 Frequency Based DME

3.2.16 Width Tuning

3.2.17 MLPN

3.2.18 Error Generator

3.2.19 PRBS Checker/Generator

4 Networking and Stateless Offloads

Note: This chapter is not relevant for Connect-IB.

This chapter covers the networking services provided by the device. Networking services include mechanisms for sending Ethernet frames over Ethernet media, Ethernet frames over InfiniBand (EoIB) and IP frames over InfiniBand (IPoIB). Networking services also enable sending Ethernet frames encapsulated over overlay networks. The overlay network can be NVGRE or VXLAN networks over Ethernet, IPoIB or EoIB. Networking services provide stateless offloads mechanisms.

The chapter is divided into the following sections:

- “[Networking Transport Objects](#)” - describes the transport objects used for sending and receiving Ethernet, IPoIB and EoIB
- “[Networking Services Usage](#)” - describes how to use networking services for Ethernet, IPoIB and EoIB
- “[Stateless Offloads](#)” - describes the various stateless offloads that are supported for Ethernet services, including tunneling that is implemented by Hypervisors
- “[Self-Loopback Control Using Transport Domains](#)” - describes provisioning of networking services of Virtual Functions, where the Hypervisor controls the underlying tunnels and the VF is unaware of it.

4.1 Networking Transport Objects

4.1.1 Overview

The adapter device uses the following transport objects (detailed in [Chapter 8, “Software Interface” on page 248](#)) for processing ingress/egress traffic:

- Flow Table - See [Section 8.18](#)
- Transport Interface Send (TIS) - See [Section 8.13](#)
- Transport Interface Receive (TIR) - See [Section 8.12](#)
- Send Queue (SQ) - See [Section 8.16](#)
- Receive Queue (RQ) - See [Section 8.14](#)
- Receive Memory Pool (RMP) - See [Section 8.17](#)
- Completion Queue (CQ) - See [Section 8.19](#)
- Event Queue (EQ) - See [Section 8.20](#)

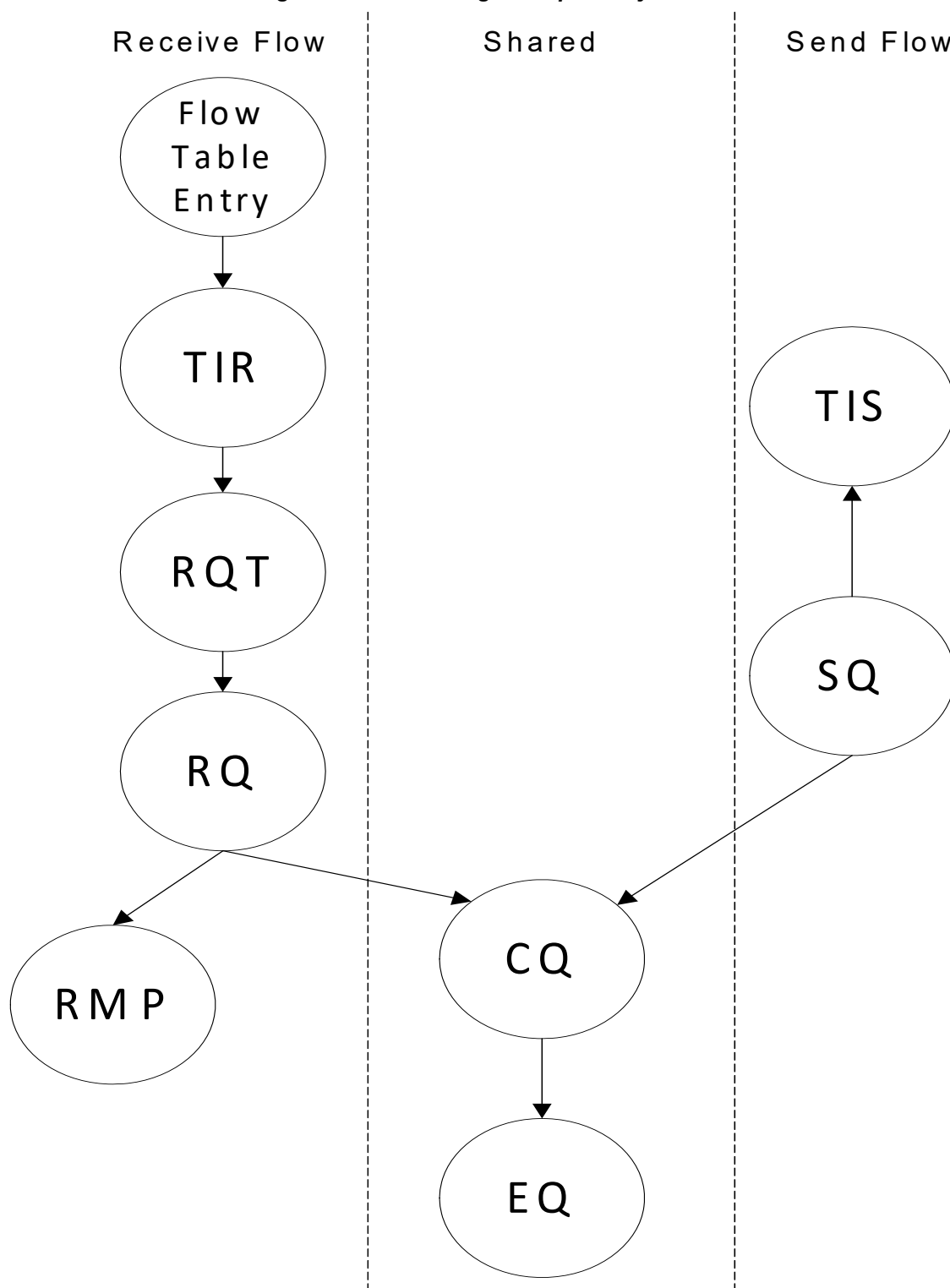
[Figure 11](#) summarizes the various objects and the relationships between them. An arrow denotes an object pointing to n other objects whereas the n is specified on the edge of the arrow.

Note: SQ and RQ can point to the same or to a different CQ.

[\[EditorNote\] - Add diagram which include only RQ and TIR for golan.](#)

[EditorNote] - Add diagram which include only RQ and TIR for golan.

Figure 11: Networking Transport Objects



As the diagram is shown for simplicity, it is imperative to explain the many to many relationships that are beyond the diagram. For example - two TIRs may point to the same or different RQs and two completely different RQs or SQs (from same or other TIS/TIRs) can point to the same CQ or the same RMP.

4.2 Networking Services Usage

4.2.1 Ethernet

[\[TBD\] TODO: add diagram for common case](#)

The basic Ethernet driver flow is summarized in the following bullets:

- Driver load
 - Check device capabilities for Ethernet offload support through the command QUERY_HCA_CAP.
 - Establish transmit rings:
 - Allocate interrupts, create Event Queues and Completion Queues. Usually one interrupt, one Event Queue and one Completion Queue are created per CPU core.
 - Create Transport Interface Send (TIS) objects. Usually one TIS is created per priority.
 - Create Send Queues (SQ) objects for sending packets and associate with TIS. Usually, Send Queues are allocated per CPU, per priority. Note that this implies a many to one relationship between SQs and CQs.
 - Establish receive rings:
 - Allocate interrupts, create Event Queues and Completion Queues. Usually, one interrupt, one Event Queue and one Completion Queue is created per CPU core.
 - Create Receive Queues (RQ) objects for receiving packets and associate with TIR (or RQT in case of indirection usage, then TIR points to RQT which point to RQs). Usually, Receive Queues are allocated per CPU.
 - Create Transport Interface Receive (TIR) objects. Associate TIR objects with the respective RQ objects. [\[TBD\] TIR is required for each Multicast group!](#)
- Flow Table:
 - Configure the Flow Table to point to the relevant TIR objects.
- Transmit Data Path:
 - Post send Work Request on the TX Work Queue buffer and ring doorbell.
 - Buffers can be released when CQE is reported.
 - Interrupt and event reporting can be controlled through the CQ and the EQ.
- Receive Data Path:
 - Post receive Work Request on the RX Work Queue buffer and ring doorbell.

- Reported CQEs indicate incoming packets including stateless offload indications.
- Interrupt and event reporting can be controlled through the CQ and the EQ.
- Driver tear-down:
 - Remove Flow Table entries.
 - Destroy receive rings:
 - Destroy RQs, RQTs and TIRs.
 - Destroy the associated CQs, EQs and interrupts.
 - Destroy transmit rings:
 - Destroy TIS objects and SQs.
 - Destroy the associated CQs, EQs and interrupts.

4.3 Stateless Offloads

The device supports multiple Stateless Offloads for Ethernet and for InfiniBand (EoIB and IPoIB). The Stateless Offloads listed in this section refer to offloads for device drivers running on bare metal device. The offloads are also applicable 'for the case of tunneling - for Hypervisors that implement the tunneling logic in software and hand over fully encapsulated frames to the hardware. Stateless Offloads include the following capabilities:

- Checksum Offload
- Large Send Offloads
- Receive Side Scaling
- Transmit Side Scaling
- Interrupt Moderation
- Large Receive Offloads
- VLAN insertion and stripping
- Flow Steering at layers 2, 3 and 4
- **Header Data Split**
- Tunneling using NVGRE¹, VXLAN², GENEVE³
 - **[internal] We additionally support encapsulation over GRE**
- Packet padding (TX)

Start of Packet Padding (RX) and End of Packet Padding (RX) adapter stateless offloads are supported for the following scenarios:

- IPv4 and IPv6 packets

1. NVGRE - IETF draft-sridharan-virtualization-nvgre-04

2. VXLAN - IETF draft-mahalingam-dutt-dcops-vxlan-09

3. GENEVE - IETF draft-gross-geneve-00. GENEVE encapsulation is supported on Eth only.

- Layer 2: Ethernet, EoIB and IPoIB

[\[TBD\] is there checksum validation on UDP on RX for outer header? - there is. Need to fix table](#)

4.3.1 Checksum Offload

The device supports calculation of checksum on transmitted packets and validation of received packets checksum. The adapter device offloads IPv4 checksum (L3) and TCP/UDP checksum (L4). Checksum calculation is supported for TCP/UDP running over IPv4 and IPv6.

Checksum offload support is reported through the *csum_cap* bit. See [Section 29.3.4.5, “Networking Offload Capabilities,” on page 1734](#).

Transmit checksum offload is supported through WQE checksum bits as described in [Section 8.9.4.1, “Send WQE Format,” on page 310](#). Note that for TCP/UDP, the device does not require any pseudo header checksum calculation, and the value placed in TCP/UDP checksum is ignored by the device when performing the calculation.

Note that UDP/TCP transmit checksum calculation is not supported for IPv6 packets with routing extension header and for IPv4 packets with LSRR and SSRR IPv4 options.

Receive checksum offload is reported through CQE checksum bits as described in [Section 8.19.1.1, “CQE Format,” on page 435](#). The relevant CQE checksum bits are:

- *l4_ok, l3_ok* - indicate whether checksum calculation of the packet(or inner packet on tunneled packets) was okay
- Checksum - this field holds 1's complement 16-bit sum of the part of the packet and is summarized in [Table 6](#). It helps software to calculate checksum on the packet in exceptional cases without the need to checksum all the packet payload by only calculating incrementally.

There are several types of packet checksum calculations performed by the device:

- IP checksum - Checksum of the IP header and IP Payload
- TCP LRO checksum - Checksum of the merged TCP segment payload
- Packet checksum - Checksum of the entire packet excluding first 14B (Ethernet), or 4B (IPoIB) and any byte that was stripped from the packet

Table 6 summarizes the Checksum field behavior in the CQE. Note that tunneled packets for Hypervisor offloads are indicated with tunneled=1. Tunneled packets for SRIOV are indicated with tunneled=0 and the offloads refer to inner frame fields.

Table 6 - CQE Checksum Behavior

CQE Indication			Checksum Field	Comment
l3_hdr_type	Tunneling	LRO Performed		
Ipv4/ Ipv6	no	no	If HCA_CAP.cqe_checksum_full==0, IP checksum. Otherwise, Packet checksum.	IP packet (IP only, TCP/UDP ...)
Ipv4/ Ipv6	no	yes	TCP LRO checksum	TCP LRO message
all	yes	no	If HCA_CAP.cqe_checksum_full==0, IP checksum (outer packet). Otherwise, Packet checksum (outer packet). Note: l3_hdr_type in CQE refers to inner packet.	Tunneled message
Ipv4/ Ipv6	yes	yes	TCP LRO checksum for the inner TCP	Tunneled TCP LRO message
None	no	no	If HCA_CAP.cqe_checksum_full==0, this field is invalid. Otherwise, Packet checksum.	Unrecognized packet, non IP

4.3.2 Large Send Offload (LSO)

The adapter device supports Large Send Offload on transmitted TCP packets over IPv4 and IPv6. Large Send Offload enables the software to prepare a large TCP message for sending with a header template which is updated automatically for every generated segment. The device segments the large TCP message into multiple TCP segments. Per each such segment, device updates the header template accordingly, and then transmits these segments into the wire. At the end, a single completion (CQE) can be reported. The device's LSO is compatible with NDIS LSOv2 specification.

Large Send offload support is reported through the QUERY_HCA command through the *max_lso_cap* field.

To activate LSO for large TCP segment, SW must follow:

- Inline in ETH Segment headers till the last byte of the TCP header.
- Set Maximum segment size in ETH Segment: EthSegment.mss

Set opcode in Ctrl Segment: CtrlSegment.OPCODE = 0x0E '01110' - Large Send Offload (LSO). Large Send Offload is supported through WQE Large Send Offload opcode (see [Section 8.9.4.1.1, "Ctrl Segment," on page 312](#)).

4.3.3 Software Parsing - SWP

Software parsing (SWP) is a generic feature that can be used by software to instruct the device to stop using its own internal parser, to parse packets on the transmit path and start parse packets according to the offsets set for each packet.

Through this feature, the device allows the handling of checksum and LSO by the HW according to the location of IP and TCP/UDP headers provided by the software.

The device does not enforce the existence of the fields indicated by the SWP feature. That is, software can choose to set each one of the `swp_*` fields in the Ethernet Segment of the WQE (See Table 47, “Eth Segment Format,” on page 318). If one of the fields is set, SWP will be enabled and Checksum/LSO will be calculated according to all offset fields.

Note:

- SWP feature is supported when `HCA_CAP.Networking_Offloading_Capabilities.swp` is set and only for SQs which `SQ.allow_swp==1`.
- SWP Checksum feature is supported when `HCA_CAP.Networking_Offloading_Capabilities.swp_csum` field is set.
- SWP LSO feature is supported when `HCA_CAP.Networking_Offloading_Capabilities.swp_lso` field is set.
- Match on flow tables will always be performed depending on Hardware parsing and not Software parsing.

4.3.4 Multi-Packet Send WQE Offload

Note: This feature is supported only when `HCA_CAP.multi_pkt_send_wqe==1` (see Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734).

Multi packet WQE enables sending multiple fix sized packets using a single WQE. The posted WQE contains pointers to the packets. Hardware breaks into packets according to `WQE.ETH_Segment.MSS` field in WQE. SW must ensure that the size of each packet is equal to this field. When WQE is completed, a single CQE will be generated.

Multi packet send WQE can be posted only to SQ in which `SQ.allow_multi_pkt_send_wqe==1` (see Table 138, “SQ Context Format,” on page 404).

Note that multi packet WQEs with zero inline headers can be supported regardless of the minimal required inline header that is set in SQ context.

4.3.4.1 Multi-Packet Send WQE Format

Multi packet send WQE format has same format as LSO WQE with the below characteristics:

- `CTRL_SEGMENT Opcode == Send LSO`.
- `CTRL_SEGMENT OpcodeMod == 1` which indicates multi packet send WQE.
- `ETH_SEGMENT.Inline_header_size == 0`.
- `ETH_SEGEMNT.mss`: in case of multi packet WQE, this field corresponds to packets fixed size.

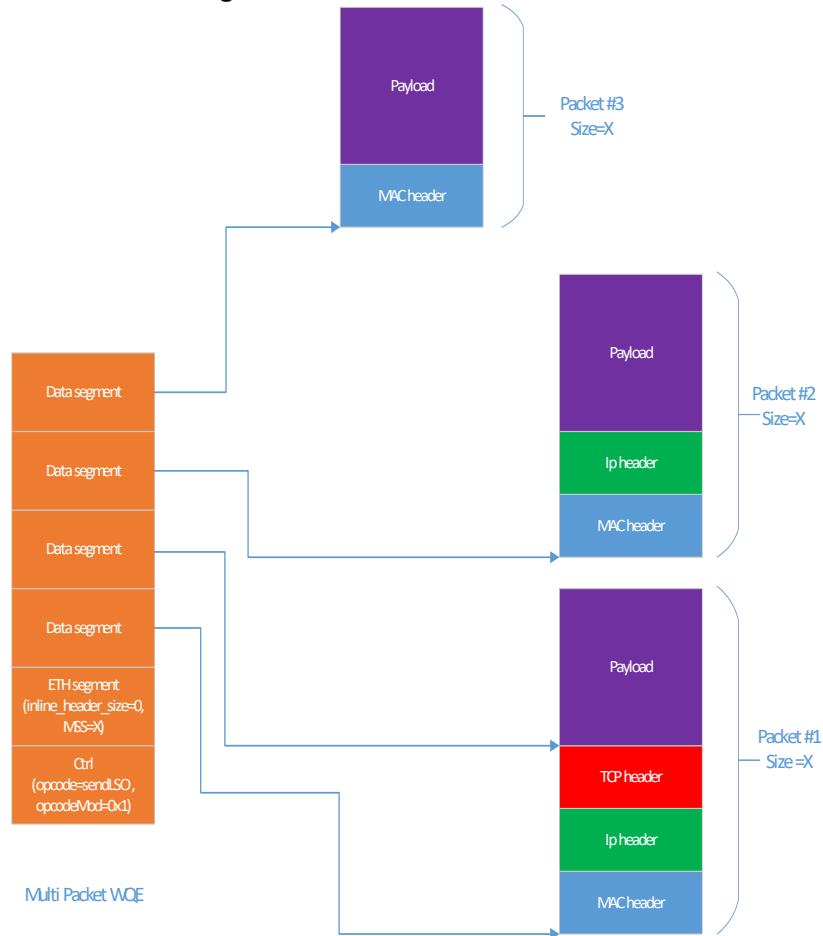
- `ETH_SEGMENT.Checksum (inner/outer)`: in case of multi packet WQE, this field applies to all packets in the WQE. That is, the NIC will try to calculate the requested checksums for each packet in the WQE.

4.3.4.2 Multi-Packet Send WQE Limitations

The following features are not supported when using multi packet send WQE:

- ACLs
- WQE Inline headers
- SRIOV (eSwitch offloads)
- priority and dscp forcing
- Loopback decision.
- vlan insertion
- encapsulation (encap/decap)
- sniffer
- LSO
- Signature

Figure 12: Multi-Packet Send WQE



4.3.5 Enhanced Multi-Packet Send WQE Offload

Note: This feature is supported only when `HCA_CAP.enhanced_multi_pkt_send_wqe==1` (see Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734).

Enhanced Multi Packet Send WQE enables sending multiple various sized packets using a single WQE, where each data segment in the posted WQE points to a separate packet.

Note that for enhanced multi-packet send WQE there is no requirement that `SQ.allow_multi_send_wqe==1`. Note also that MTU is enforced on each packet of the WQE individually.

4.3.5.1 Enhanced Multi-Packet Send WQE Format

Enhanced multi-packet send WQE format:

- `CTRL_SEGMENT Opcode == ENHANCED_MPSW`.
- If `ETH_SEGMENT` exists, then it is shared for all the packets and `ETH_SEGMENT.Inline_header_size` must be 0.

- Each `DATA_SEGMENT` define a separated packet. Note that if data segment is inline with `ByteCount=0x0`, then it is silently skipped by the device (will not yield any packet). This property may be used by the driver to align one or more inline packets of a multi-packet WQE to the system cache line.

4.3.6 Receive Side Scaling (RSS)

The adapter device supports Receive Side Scaling (RSS) capability. RSS spreads incoming traffic across multiple rings which can be further redirected to different CPU cores to better spread incoming loads. The adapter's RSS is compatible with the NDIS RSS specification.

RSS is implemented through two mechanisms: Flow Table and TIRs. Flow Table rules classify packets and deliver a specific traffic classes (e.g. TCP/UDP, IP only) to dedicated TIRs. TIRs of type Indirect are responsible for spreading the traffic which they receive from Flow Table and which belongs to specific traffic classes. Default RSS RQ can be supported by opening a dedicated TIR of the type Direct associated with a default RQ and configuring Flow Table to deliver to that "default" TIR all the traffic classes which should be delivered to the default RQ instead of being spread.

This allows supporting different RSS spreading modes for different traffic classes. For example, Flow Table rules can deliver all TCP IPv4 packets to TIR which spreads using <Source IP, Destination IP, Source TCP Port, Destination TCP port>, deliver TCP IPv6 packets to TIR which spreads using <Source IP, Destination IP> and deliver UDP packets to the "default" TIR.

For more details, please refer to [Section 8.18, "Flow Table," on page 414](#).

The actual support for RSS is reported through the `HCA_CAP.rss_ind_tbl_cap`.

4.3.6.1 TIR Spreading Traffic Mechanism

TIR context, with dispatcher type Indirect, selects a destination RQ based on selected packet fields. The TIR uses a configurable hash function and calculates a hash value based on the selected packet fields. TIR context selects a packet field according to `rx_hash_fields_selector_outer` and `rx_hash_fields_selector_inner` fields that set which field must take part in RQ selection. Note that only packets which include the fields chosen by TIR's `rx_hash_fields_selector_outer/rx_hash_fields_selector_inner` must be delivered to this specified TIR. For example, if TCP destination/source port are selected, only packets with TCP destination/source ports (UDP/TCP) should be delivered to that TIR. This can be achieved via proper Flow table rule configuration, which will ensure, for example, that only TCP/UDP packets are delivered to the TIR which uses TCP ports for RQ selection. The calculated hash value points to an indirection table (implemented with RQs Tables). The indirection table entries point to RQs for further handling of incoming packets. The RX hash value, selected fields and packet types are reported to the CQE. For more details related to TIR object, refer to [Table 117, "TIR Context Fields," on page 376](#).

4.3.7 Transmit Side Scaling

Transmit side scaling enables simultaneous transmission of packets on multiple descriptor rings from multiple CPU cores. To achieve this, more than one SQ should be created. Multiple SQs can be attached to the same TIS.

[\[TBD\] Capability bit](#)

4.3.8 Interrupt Moderation

Interrupt moderation can be achieved by controlling the EQ generation by CQs. This mechanism is explained in detail in [Section 8.20.19, “Completion Event Moderation,” on page 508](#).

[\[TBD\] Capability bit](#)

4.3.9 Large Receive Offload (LRO)

4.3.9.1 LRO Introduction and Device Capabilities

The device supports Large Receive Offload (LRO) for TCP streams. The implementation is fully compatible with NDIS Receive Segment Coalescing (RSC).

LRO capabilities can be queried through the `QUERY_HCA_CAP` command and is reflected through the following bits (see [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667](#)):

- *lro_cap* - indicates LRO support
- *lro_psh_flag*, *lro_time_stamp* - indicate LRO support for segments with PSH flag and with the TCP timestamp option
- *lro_min_mss_size* - the minimal size of TCP segment required for coalescing
- *lro_timer_supported_periods* array - lists the supported LRO timer periods

LRO can be configured per TIR when the TIR is created through the `CREATE_TIR` command. The LRO parameters can be dynamically updated on a TIR using the `MODIFY_TIR` command, which can also turn on and turn off the LRO.

When a TIR that is enabled for LRO is destroyed (`DESTROY_TIR`), the device first completes all outstanding LRO sessions on that TIR and generates completions (CQEs). Only then does the `DESTROY_TIR` command complete. Similarly, when modifying a TIR (`MODIFY_TIR`) to disable LRO, the device first completes all outstanding LRO sessions on the TIR.

When enabled for LRO, the TIR extracts LRO Flow IDs for incoming packets. LRO Flow ID is based on connection 6 tuple: VLAN ID, Source IP, Destination IP, Source TCP Port, Destination TCP Port and TCP transport. Packets that use the same LRO Flow ID are candidates for LRO coalescing.

4.3.9.2 LRO Session Creation

LRO coalescing is performed using LRO sessions. Sessions can be created and terminated dynamically by the device. While a session is open, segments using the same LRO Flow ID can be coalesced within the session.

A session is created when all the following conditions are met:

- LRO is enabled when creating TIR context by setting *lro_enable_mask* and other lro parameters
- TCP Packet

- No LLC/SNAP
- The HW has available resources for a new LRO session [\[TBD\] Quota?](#)
- Packet validation is OK, e.g. packet is not malformed, TCP/IP checksum is OK, etc.
- TCP flags are clear: SYN, URG, RST, FIN, NS and Reserved flags
- When HCA_CAP.lro_psh_flag is disabled, PSH flag should also be clear
- No TCP options other than timestamp
- When HCA_CAP.lro_time_stamp is disabled, TCP timestamp option must not appear
- No IPv4 options or IPv6 extension headers
- The packet is not an IP fragment
 - [internal] This is configured via Chickenbit, IMO LRO with SNAP/LLC should be disabled since Linux OS may not digest this well in Window supporting it is not required and this is not a very interesting and important
- Segment TCP payload size is $\geq$ HCA_CAP.lro_min_mss_size
 - Note: lro_min_mss_size > 0 and therefore pure ACKs are never merged
- LRO Flow ID does not have an open LRO session

4.3.9.3 LRO Session Termination

An LRO session is terminated when one of the below conditions is met:

- LRO timer expiration
- Not enough space in receive WQE
- TIR destruction or update (e.g. LRO disable)
- Incoming packets that belong to the LRO session and meet any of the conditions which are described below:
 - Invalid TCP/IP checksum
 - One of the flags is enabled: SYN, URG, RST, FIN, NS and reserved TCP flags
 - PSH flag is enabled and HCA_CAP.lro_psh_flag = 0
 - The packet contains TCP options other than timestamp
 - When HCA_CAP.lro_time_stamp is disabled and TCP timestamp option appears
 - The packet contains IPv4 options or IPv6 extension headers
 - Segment TCP payload size is $<$ HCA_CAP.lro_min_mss_size
 - Out of order segment (TCP sequence number doesn't match the expected sequence number)
 - Coalescing the currently arrived segment will exceed TIR's maximum IP length(*)

- Segment contains a timestamp option, `HCA_CAP.lro_time_stamp == 1`, and one of the following conditions is met(*):
 - The previous segment does not contain the timestamp option
 - Timestamp Value of the new segments < Timestamp Value of the LRO session
- The segment does not contain a timestamp option but the previous segments contain a timestamp option (*)
- The segment contains a different value for the ECN flags (ECE and CWR) in the TCP header. The segment contains a different value for the ECN field (ECT, CE) in the IP header comparing to the previous segment (*)
- IP field of the incoming packet does not match the IP fields of the previous packets besides the following fields: TTL/Hop Limit, IPv4.Checksum, IPv4.Total Length and IPv4.Identification/IPv6.Payload Length
- The packet is an IP fragment.

When the session is terminated, a completion is generated accordingly to reflect the data that was already coalesced on the LRO session. When session termination happens as a result of segment arrival, this segment is not included as part of the “already coalesced data.” After the completion generation, the new segment can be a candidate for a new session or generate an immediate completion. If the session was terminated as a result of timestamp mismatch or ECN mismatch (last four cases of incoming packet, marked with asterisk (*) in the above list), then the new segment can be a candidate for creation of a new session (see [Section 4.3.9.2, “LRO Session Creation,” on page 195](#)). Otherwise, the new segment generates a regular completion as a standalone incoming packet.

4.3.9.4 Merging New Segments Into Existing LRO Session

When a new segment arrives with an LRO Flow ID of an existing LRO session, then the segment is a candidate for coalescing into an LRO session. The merge takes place if all the below conditions are met:

- All conditions mentioned in [Section 4.3.9.2, “LRO Session Creation,” on page 195](#)
- If the segment contains a timestamp option and:
 - the previous segment contained a timestamp option
 - the incoming segment timestamp value is $\geq$ the timestamp previous segment timestamp
- If the incoming segment does not contain a timestamp option and the previous segment did not contain a timestamp option
- Matching on all IP fields, except: TTL/Hop Limit, IPv4.Checksum, IPv4. Total Length and IPv4.Identification/IPv6.Payload Length
- Matching on ECN flags (ECE and CWR) in the TCP header
- Matching on TCP Sequence Number
- Coalescing the new segment will not exhaust `TIR lro_max_ip_payload_size`

- Please note that the RQ WQE should be large enough to contain the merged segments, otherwise the LRO session is flushed and completion with error is generated.

[\[TBD\] Discuss with Alex whether the reason for not merging means that we need to terminate](#)

4.3.9.5 LRO Packet Scatter to Memory

LRO session is scattered into memory according to the relevant RQ/RMP pointed to by the TIR. The first segment in the LRO session is scattered as is to memory according to the RQ configuration. For example, when VLAN stripping is configured, then the VLAN is stripped prior to scattering the first segment into memory. CQE is not reported. Subsequent segments that are merged into an opened LRO session continue to utilize that WQE. Therefore that WQE has to be large enough to contain the `lro_max_ip_payload_size` (as configured in the TIR) and in addition the IP headers and the L2 header.

Subsequent segments that are added to the LRO WQE are adding only the TCP payload, thus creating a big TCP packet (one header with coalesced TCP payload) that comprises all coalesced segments. The segments do not update the TCP header of the first segment (the one that created the LRO session). The relevant TCP fields, e.g. latest TCP Window Size, are reflected in the CQE that is generated when the LRO session is terminated.

It is the software responsibility to fix the original TCP and IP headers based on the CQE values. To do this, software can check the `lro_num_seg` in the CQE and if non-zero, then TCP fields should be updated.

Note that additional information for debug, tuning and statistics is reported to the LRO CQE, e.g. the reason for the LRO session termination.

Note: RQs used for LRO-enabled TIRs should not be configured to scatter the FCS (`RQ.scatter_fcs`)

[\[TBD\] what about special offloads like end of packet padding, etc.](#)

[\[TBD\] Add a picture](#)

4.3.9.6 LRO CQE Fields Summary

This section summarizes LRO related CQE fields and their usage. SW needs some of these fields in order to update the TCP/IP headers to correctly reflect the arrived data and connection state (for example TCP Window size or IPv4 TTL field). These fields should be carefully updated according to the requirements and needs of the relevant SW framework.

Table 7 summarizes the LRO fields in CQEs.

Table 7 - LRO CQE Fields Summary

Field Name	Field Value	Description
<code>lro_tcp_psh</code>	One of the packets in the session had a TCP PUSH flag set.	If set, SW must enable the PSH flag in the header of the merged TCP segment. Valid only for LRO packets (<i><code>lro_num_seg</code></i> field not clear).

Table 7 - LRO CQE Fields Summary

Field Name	Field Value	Description
lro_abrt	Abort coalescing reason.	Can be used in order to collect LRO statistic. Valid only for LRO packets (<i>lro_num_seg</i> field not clear).
lro_dup_ack_cnt	Number of duplicate acknowledgment that were encountered while creating the coalesced segment.	In some environments driver must report this number as OOB info to TCP/IP stack. [Internal] For WIN Valid only for LRO packets (<i>lro_num_seg</i> field not clear). This field should be always zero.
lro_min_ttl	The minimum HopLimit / Time-ToLive of all packets in this merged segment.	SW must update the IPv4 Time to live or IPv6 Hop limit fields according this field. Valid only for LRO packets (<i>lro_num_seg</i> field not clear).
lro_tcp_win	TCP window of the last packet in the coalesced segment.	SW must insert this value into the TCP header of the merged packet.
lro_ack_seq_num	LRO acknowledgment sequence number (last packet in session).	SW must insert this value into the TCP header of the merged packet. Should be ignored if <i>l4_hdr_type</i> != 3 or 4. Valid only for LRO packets (<i>lro_num_seg</i> field not clear).
Checksum	The checksum field for coalesced TCP segments.	For LRO (<i>lro_num_seg</i> field not clear), this field covers TCP header + TCP payload of all merged packets (without TCP pseudo header). If SW wants/needs to calculate and fix checksum of merged TCP header and payload it can use this field. This field helps software calculate checksum on the merged packet without the need to checksum all the packet payload by only calculating incrementally. Please note that the final checksum should be calculated after TCP header is fixed. Please note that this field is not useful for IPv4 Checksum calculation. Also please note that before recalculating and fixing IPv4 Checksum IPv4 header must be first fixed so it will reflect the correctly the merged segment.
lro_num_seg	Number of coalesced segments of LRO.	0 - when LRO was not performed. 1 - when HW aggregated just a single packet. Greater than 1 - when HW aggregated more than a single packet.

Table 7 - LRO CQE Fields Summary

Field Name	Field Value	Description
byte_cnt	Byte count of data transferred.	For LRO messages (<i>lro_num_seg</i> field not clear): the total length of messages is specified. For example, for ETH packets the byte count starts from the MAC header. This field should be used in order to fix the IPv4. Total length or IPv6 Payload length fields.
lro_timestamp_is_valid	Indicates that CQE's timestamp field holds LRO timestamp instead of regular timestamp CQE field.	When this bit is set <i>lro_timestamp_value/timestamp_h</i> and <i>lro_timestamp_echo/timestamp_l</i> holds LRO timestamp values.
lro_timestamp_value/ timestamp_h	This field holds the Timestamp Value (TSval) field of the header of the last coalesced segment.	This field holds the Timestamp Value (<i>TSval</i>) field of the header of the last coalesced segment only if <i>lro_timestamp_is_valid</i> bit is on. If this field is valid (as LRO timestamp), SW can choose to update the TS Value (<i>TSval</i>) field of TCP header. Please note that this is not always required/needed. Please refer to documentation of your SW framework for more details.
lro_timestamp_echo/ timestamp_l	This field holds the Timestamp Echo Reply (TSecr) field of the header of the last coalesced segment.	This field holds the Timestamp Echo Reply (<i>TSecr</i>) of the header of the last coalesced segment only if: <i>lro_timestamp_is_valid</i> && (<i>L4_hdr_type</i> == 3 <i>L4_hdr_type</i> == 4). If this field is valid (as LRO timestamp), SW can update TS Echo Reply (<i>TSecr</i>) field of the TCP header.
L4_hdr_type	L4 Header type	In some cases for LRO messages (<i>lro_num_seg</i> > 0), inside CQE, the <ACK> bit is enabled but inside the header the bit is disabled. This happens when in the header of the first segment, the <ACK> bit, was off but in one of the next merged segments this bit was on. So for LRO packet when <i>L4_hdr_type</i> == 4 or 3, SW must verify that the <ACK> bit inside the TCP header of the merged segment is enabled, before delivering the segment to the TCP/IP module. In addition this bit allows checking if <i>lro_ack_seq_num</i> and <i>lro_timestamp_echo</i> field are valid.

4.3.10 VLAN Insertion/Stripping

The device offloads VLAN insertion and stripping for raw Ethernet frames.

VLAN insertion is performed by the driver, inlining the VLAN tag into the Ethernet frame headers in the WQE Eth Segment (see [Section 8.9.4.1.5, “Eth Segment and Padding Segment,”](#) on page 318).

VLAN Stripping is configured in RQ through the VLAN Stripping Disable (vsd) bit. When configured to perform VLAN Stripping, the device removes the VLAN tag from the incoming frames and reports it in the CQE fields.

For tunneled packets, both the outer and the inner VLANs can be manipulated through VLAN insertion offload, by inlining both outer and inner frame headers, but only the outer VLAN can be stripped by VLAN stripping offload.

The support for VLAN insertion by WQE Eth Segment and VLAN stripping is reported through the QUERY_HCA_CAP *vlan_strip* field.

When `HCA_CAP.wqe_vlan_insert==1` and `HCA_CAP.wqe_inline_mode=0x2` (inline not required), the device supports VLAN insertion into the packet without copying any other packet header to the WQE by setting *insert_vlan* field in Wqe Eth Segment. See [Table 48, “Eth Segment Fields,”](#) on page 319.

[\[TBD\] is a match must? discard on non match? multimatch is not supported only first hit, right?](#)

4.3.11 Header Data Split

[\[TBD\] Capability bit](#)

4.3.12 Hypervisor Offloading for Tunneling

Tunneling technologies, introduced for solving scalability and security challenges, require extension of the traditional stateless offloads to avoid performance drop. The adapter device offers the stateless offloads for a tunneled packet, similar to the ones offered to non-tunneled packets.

4.3.12.1 Configuration

Stateless Offloads for tunneled packets can be configured independently per offload type as described in the sections below.

Supported tunneling types can be queried via QUERY_HCA_CAP command.

Note: The device can recognize and treat an incoming tunneling packet as an encapsulated packet and support receive Stateless Offloads for the inner packet’s fields only when the packet is processed by TIR which is enabled to support tunneling. Otherwise, the device treats the packet as an “ordinary” non-tunneling packet and receive offloads can be done only for the outer packet’s fields.

Note: Receive Flow steering based on inner tunneling packet fields can be done regardless of which TIR will process the incoming packet.

In order to enable receive Stateless Offloading support for incoming tunneling traffic the following must be done:

- Enable TIR/s to support Stateless Offloads for tunneled packets. TIR is enabled to support tunneling by setting `TIR.tunneled_offload_en = 1` (see [Table 116, “TIR Context Format,” on page 374](#)).
- Configure Flow Table to deliver the tunneling packets to TIR which supports Stateless Offloading for tunneling.

Note that in order to support Stateless Offloads for VXLAN packets the VXLAN UDP port/s must be registered via `ADD_VXLAN_UDP_DPORT` command (see [Section 29.42, “VXLAN Commands,” on page 2251](#)).

4.3.12.2 Indication in CQE for Tunneling Packet

SW can recognize a tunneling packet by reading the `CQE.tunneled` field. When this field is enabled, the device identifies a tunneling packet and successfully validates the L4 or L3 Checksums of the outer headers. If outer L3 or L4 Checksum validation fails, the device treats the packet as non-tunneling, `CQE.tunneled` field is disabled and CQE fields reflect properties of outer headers.

When `CQE.tunneled` is enabled, CQE fields report IP and TCP/IP properties of the tunneled (inner) packet including checksum validation status. In addition, the following “extra” fields (in a sense that for non-tunneling they are not used) are updated by the adapter device in order to report properties of the tunneling (outer) packet: ***outer_ip_ext_opts*** and ***outer_l3_type***.

Note: ***rx_hash_result*** and ***rx_hash_type*** fields refer to outer or inner depending on incoming packet type, Flow Table rules and TIR configuration.

Note: Checksum field behavior is described in [Table 6, “CQE Checksum Behavior,” on page 190](#).

Note: ***hds_type***, ***cv***, ***up***, ***cfi*** and ***vid*** refer to tunneling (outer) packet also when `CQE.tunneled` bit is set.

4.3.12.3 RX Hash (RSS) for Tunneling Packets

Tunneling frames can be distributed by RX Hash mechanism using the inner or outer packet fields. Different spreading schemes (which inner or which outer fields to use for spreading, hash function configuration etc...) can be applied to different tunneling types. For example, the device can be configured to spread VXLAN packets using inner packet fields but can also be configured to spread NVGRE packets using the outer packet fields. In addition, the device can spread by selecting a different set of inner or outer packet fields for each tunneling type. Moreover, the device allows configuring different spreading schemes for different tunneling/tunneled flows. Flow Table and TIR configuration allow setting how the spreading is being done for the specific tunneling type, tunneling flow type or tunneled flow type.

For details regarding RX Hash, please refer to [Section 4.3.6, “Receive Side Scaling \(RSS\),” on page 194](#).

For details related to Flow Table, please refer to [Section 8.18, “Flow Table,” on page 414](#).

For details related to TIR object, please refer to [Table 116, “TIR Context Format,” on page 374](#).

4.3.12.4 Flow Rules for Tunneling Packets

Tunneling frames may be steered to dedicated TIRs according to outer as well as inner packet's headers. In order for flow rule to steer tunneling packets, it must be in the appropriate format. Each tunneling type has its unique format. Formats for each tunneling type are further described below.

VXLAN Flow Rules Format:

In order to create Flow Table rules for VXLAN packets, the following template must be used:

Outer packet header – packet fields:

- ether_type == 0x0800 (IPv4)/ 0x86DD (IPv6)
- ip_protocol = 17 (UDP)
- l4_dport = VXLAN UDP Destination Port. Must be synchronized with VXLAN UDP Ports registered via ADD_VXLAN_UDP_DPORT command.

NVGRE Flow Rules Format

In order to create Flow Table rules for NVGRE packets, the following template must be used:

Outer packet header – packet fields:

- ether_type == 0x0800 (IPv4)/ 0x86DD (IPv6)
- ip_protocol = 47 (GRE)

Misc header – packet fields:

- gre_protocol = 0x6558 (transparent Ethernet bridging)

4.3.12.5 Checksum

Checksum Calculation on TX

Inner IPv4 and inner TCP/UDP checksums are calculated by HW, according to WQE.ETHSegment.l3cs_inner and WQE.ETHSegment.l4cs_inner fields (see [Section 8.9.4.1.5, “Eth Segment and Padding Segment,” on page 318](#)). Outer IP and TCP/UDP checksum calculation is controlled by WQE's ETH Segments as for non-tunneling packets.

Note: When inner L4 Checksum calculation is requested, HW does not support outer L4 Checksum validation, so requesting in the same WQE for both outer L4 Checksum and inner L4 Checksum is not allowed.

Checksum Validation on RX:

For tunneling packets (CQE.tunneled bit is set) **CQE.l4_ok** and **CQE.l3_ok** fields indicate checksum validation status for the inner headers. The packet is considered a tunneling packet (and subsequently CQE.tunneled bit is set) only if outer L3 and if available L4 checksums are

successfully validated; otherwise, the packet is treated as a non-tunneling packet (CQE.tunneled bit is off) with a bad checksum and **CQE.I4_ok** together with **CQE.I3_ok** indicate checksum status of the outer headers.

For details related to Checksum, please refer to [Section 4.3.1, “Checksum Offload,” on page 189](#).

4.3.12.6 LSO

The device supports Large Send Offload for TCP over tunneled IPv4 and tunneled IPv6 packets. The adapters' LSO for tunneled packets is compatible with NDIS specifications.

Tunneled TCP frames are segmented by HW, by using an LSO segment inlined in the WQE's ETH Segment as a template. Outer IP, UDP headers (when available), and inner IP and TCP headers are updated accordingly.

Note that when outer UDP header exists, UDP.Checksum field must be zeroed and HW Checksum generation for the outer UDP must be disabled (by setting EthSegment.L4cs = 0). Configuration of the outer IP.identification increment policy is done by setting HCA_CAP.tunnel_lso_const_out_ip_id field.

In summary, SW must perform the following:

- Control Path:
 - Configure the increment policy of the IP.identification: set Nic_Vport.tunnel_lso_out_ip_id_no_incr.
- Data Path:
 - Inline in ETH Segment headers till the inner TCP header.
 - Set Maximum inner segment size in ETH Segment: EthSegment.mss
 - Disable generation for the outer UDP Checksum, if outer UDP header exists, by setting EthSegment.L4cs = 0 (Checksum field of the outer UDP header must be zero)
 - Set opcode in Ctrl Segment: CtrlSegment.OPCODE = 0x0E '01110' - Large Send Offload (LSO)

Note: MSS value for LSO WQE should refer to the tunneled (inner) frame; adjustment for the MTU may be required to consider encapsulation headers.

Note: If SW does not inline till the inner TCP header, LSO will be done for the outer TCP header if appears. **[Internal] In STT we have a TCP like header.**

For more details related to LSO, please refer to [Section 4.3.2, “Large Send Offload \(LSO\),” on page 190](#).

4.3.12.7 VLAN Stripping/Insertion

For tunneled packets, both the outer and the inner VLANs can be manipulated through VLAN insertion offload, by inlining both outer and inner frame headers. However, only the outer VLAN can be stripped by VLAN stripping offload.

For more details related to VLAN Stripping/Insertion, please refer to [Section 4.3.10, “VLAN Insertion/Stripping,” on page 201](#).

4.3.12.8 Header Data Split

Header Data Split for tunneling packets is done only for the outer headers as for non-tunneling packets.

4.3.12.9 LRO

TBD

4.3.13 Packet Padding (TX)

Ethernet packets shorter than 64 bytes are automatically padded by the device on transmission to 64 bytes by appending trailing zeros (before the FCS). Additionally, the last segment of an LSO packet, if shorter than 64B, is also automatically padded to 64B.

For encapsulated frames over Ethernet, the outer Ethernet packets are automatically padded by the device to 64 bytes. Inner frames are left untouched.

4.3.14 Start of Packet Padding (RX)

Note - This feature is supported only when `HCA_CAP.start_pad==1`.

Some root complex are able to optimize their performance when incoming data size is multiple full cache lines. The device can pad the beginning of incoming packets to full cache line such that the first upstream write generated by the incoming packet will be a full cache line.

Start of packet padding can be enabled by setting *start_padding* field in Data Segment of Receive WQE to 1. See [Section 8.9.4.1.16, “Receive Data Segments,” on page 341](#).

4.3.15 End of Packet Padding (RX)

Note - This feature is supported only when `HCA_CAP.end_pad==1`.

Some root complex are able to optimize their performance when incoming data is multiple full cache lines. The device can pad the ending of incoming packets to full cache line such that the last upstream write generated by the incoming packet, will be a full cache line. When padding incoming packets, the device will never go beyond the WQE scatter entry length.

To configure QP/RQ to pad incoming packets, the *end_padding_mode* field should be configured in the QP/RQ Context.

The end of packet padding byte count is not reported to the CQE generated. In other words, the *byte_cnt* field in the CQE reports the incoming message as received bytes and excludes the padding.

Cache line size, by default, is 64 bytes and can be configured to 128 bytes using `HCA_CAP.cache_line_128byte` in case it is supported.

[\[TBD\] this was removed from Shomron POR](#)

4.3.16 Trailer Insertion

The device supports adding Trailers to Ethernet frames. The trailer is added according to the field `insert_trailer` of the Eth Segment of the WQE. Although the trailer is always added at the end of the frame, it may be associated with a number of different headers. For example, the trailer could belong to the IP layer of a TCP packet, meaning the trailer does not belong to the TCP payload. In order to perform the correct packet modifications (for example packet length, checksum calculation):

- the field `trailer_header_association` in the Eth segment should indicate the header to which the trailer belongs. Following headers (more inner) will not be modified due to the insertion of the trailer
- The packet modification should be explicitly requested (if applicable). For example, for IPv4 checksum calculation, the `l3cs` field in Eth segment should be set.

The trailer data should be placed in the first Send Data segment of the WQE. This segment must be inline (`is_inline = 0x1`).

For LSO offload, a trailer will be added for every segment of the TCP stream.

Note: If the Trailer harms the packet alignment (e.g. 4B Ethernet alignment requirement), the Trailer should be padded to accommodate. For LSO it may mean that the last segment may require a different padding. In such cases, the TCP message should be broken into more than one WQE, so that each will handle a different alignment. For example, first WQE handles all the MSS sized segments, and the second WQE handles the last segment.

The support for Trailer insertion is indicated by `insert_trailer` field in `QUERY_HCA_CAP` under Per Protocol Networking Offload Capabilities.

4.4 Self-Loopback Control Using Transport Domains

Multiple consumers that coexist on the same NIC port can communicate via a local loopback mechanism. A consumer might want to send packets to other local consumers via loopback but block self-loopbacked packets. For this purpose, the device provides Transport Domain support which allows a consumer to define which set of TIS/TIRs belongs to it and control if local loopbacked packets generated by its TISs and delivered to its TIRs should be accepted or blocked. The Transport Domain support is indicated by `HCA_CAP.log_max_transport_domain` (see [Table 1571, “HCA Capabilities Layout,” on page 1671](#)).

Transport Domains are allocated and released using `DE/ALLOC_TRANSPORT_DOMAIN` commands.

TIRs and TISs are associated with Transport Domain via `TIR/TIS.transport_domain_id` field and `TIR.self_lb_en` controls self-loopback behavior of the TIR.

4.5 Sniffer

The device deploys a sniffer mechanism, enabling the duplication of the packets to designated NIC root Flow Tables: one root Flow Table of type “NIC Sniffer RX” sniffs NIC’s incoming packets and one root Flow Table of type “NIC Sniffer TX” sniffs NIC’s sent packets. Sniffer root Flow Tables allow to create additional Flow Table chaining hierarchy, independent from the “main” NIC Flow Table configuration. This allows the Sniffer SW to manage its own Flow Table rules which are not dependent on the “original” Flow Table settings. In a sense, the sniffer is a silent “observer”; all packets are still delivered to their “original” destination when the sniffer function is enabled.

Sniffer Flow Tables must be handled as any other NIC Flow Table. Sniffer Flow Tables are managed via `CREATE/DESTROY_FLOW_TABLE` commands. See [Section 29.18, “Flow Table Commands,” on page 1985](#).

If root Sniffer Flow Tables are not configured, HW does not perform packet replication for sniffing purpose.

Note: Enabling sniffer Flow Tables may have an adverse effect on the performance of the NIC.

Note: Loopbacked-by-NIC packets are delivered both to NIC TX Sniffer Flow Table and to NIC RX Sniffer Flow Table; i.e, when both RX and TX Sniffer Flow Tables are enabled, two copies of the loopbacked packet are captured by the Sniffer.

Figure 13: Sniffer Flow in ETH Port

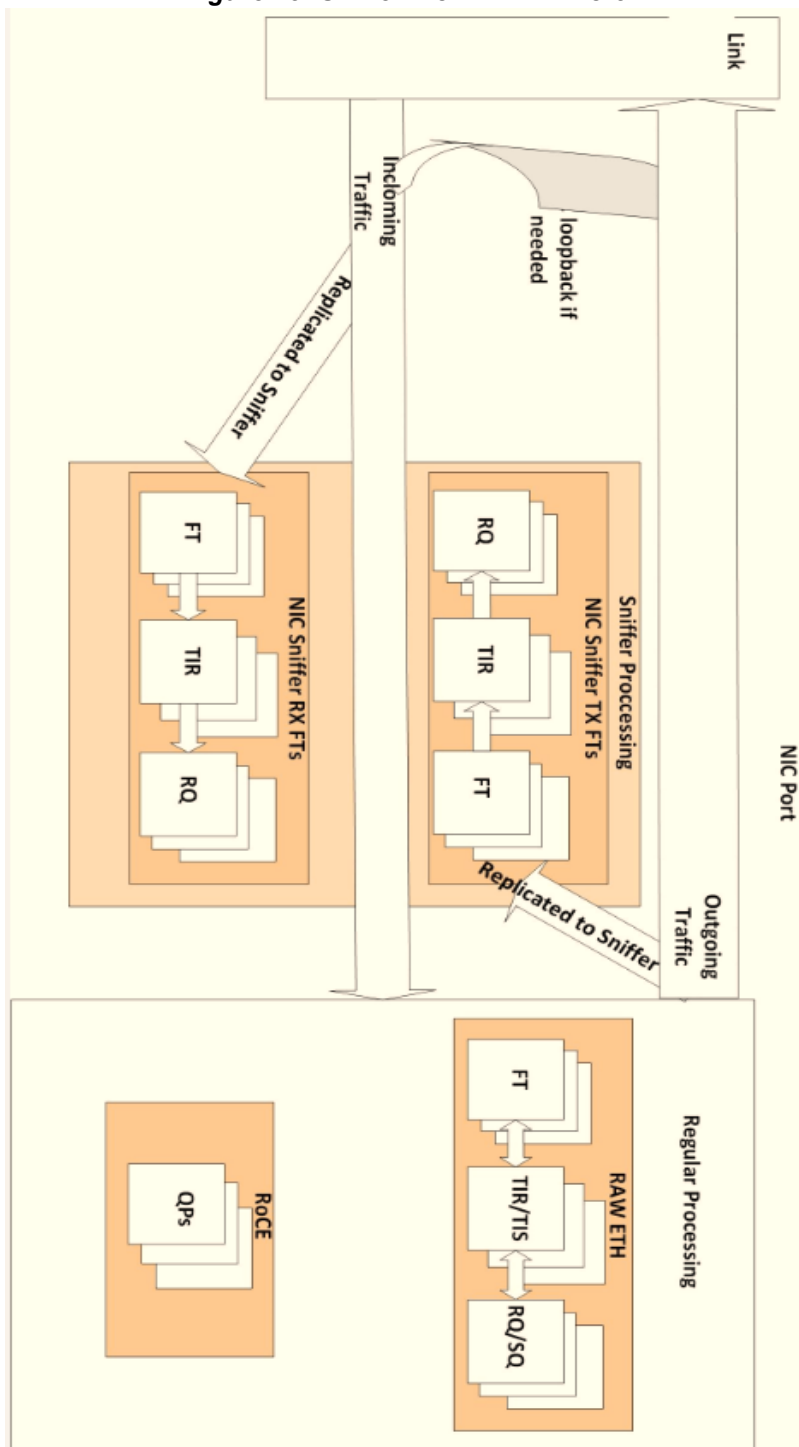
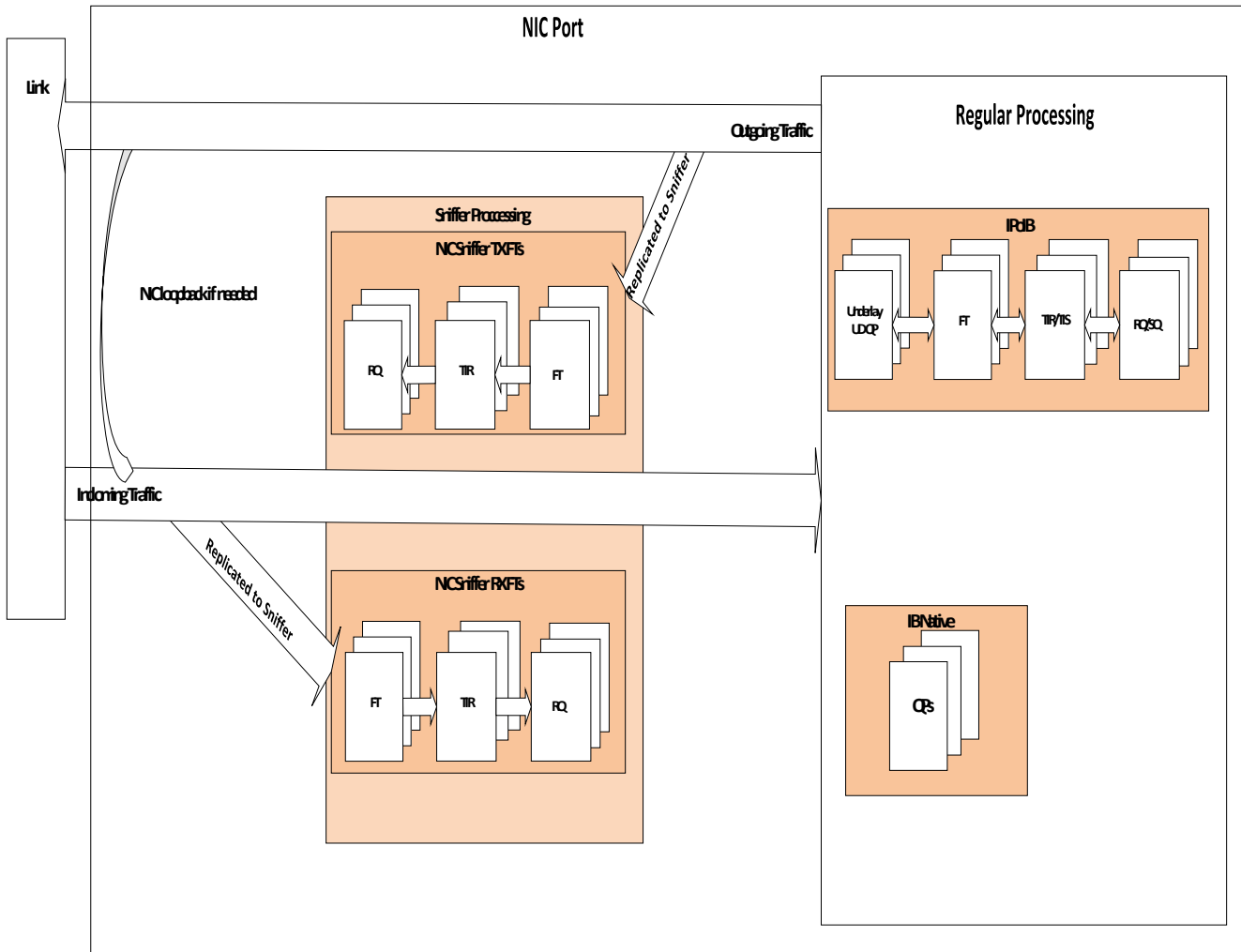


Figure 14: Sniffer Flow in IB Port



4.6 ULP Stateless Offloads

UD QPs can support multiple Stateless Offloads for IPoIB/EoIB Upper Layer Protocols - for example, TCP/IP checksum generation and validation or RSS based on TCP/IP headers and many more.

The device supports multiple ULP Stateless Offloads modes. For more information about supported ULP offloads and modes, see [Section 4.6.1, “ULP Stateless Offloads QP Modes,”](#) on page 209.

4.6.1 ULP Stateless Offloads QP Modes

4.6.1.1 Basic ULP Offloads

A QP in basic mode can support basic Stateless Offloads for the overlay part of the ULP packet.

In this mode, the ULP offloads are limited to those that are performed by ULP RQ and ULP SQ. Thus, for example, steering that is performed by Flow Table and RSS offload that is performed by TIR are not supported.

In this mode, there is no need to allocate any “Networking Transport Objects” (Flow Tables, TIRs, TISs, RQ, SQ) but only QP. Thus, utilizing basic offloads is sufficient as it requires simple control flows.

This mode is supported when `HCA_CAP.X_basic_offload==1`, where X is the ULP like IPoIB.

To enable a specific mode, SW should set it using `QPC.ulp_stateless_offloads_mode = 1` when creating a CP. See [Table 19, “QP Context Structure Layout,” on page 258](#).

4.6.1.2 Enhanced ULP Offloads

A QP in enhanced mode can support enhanced Stateless Offloads (for example steering and RSS) for the overlay part of the ULP packet.

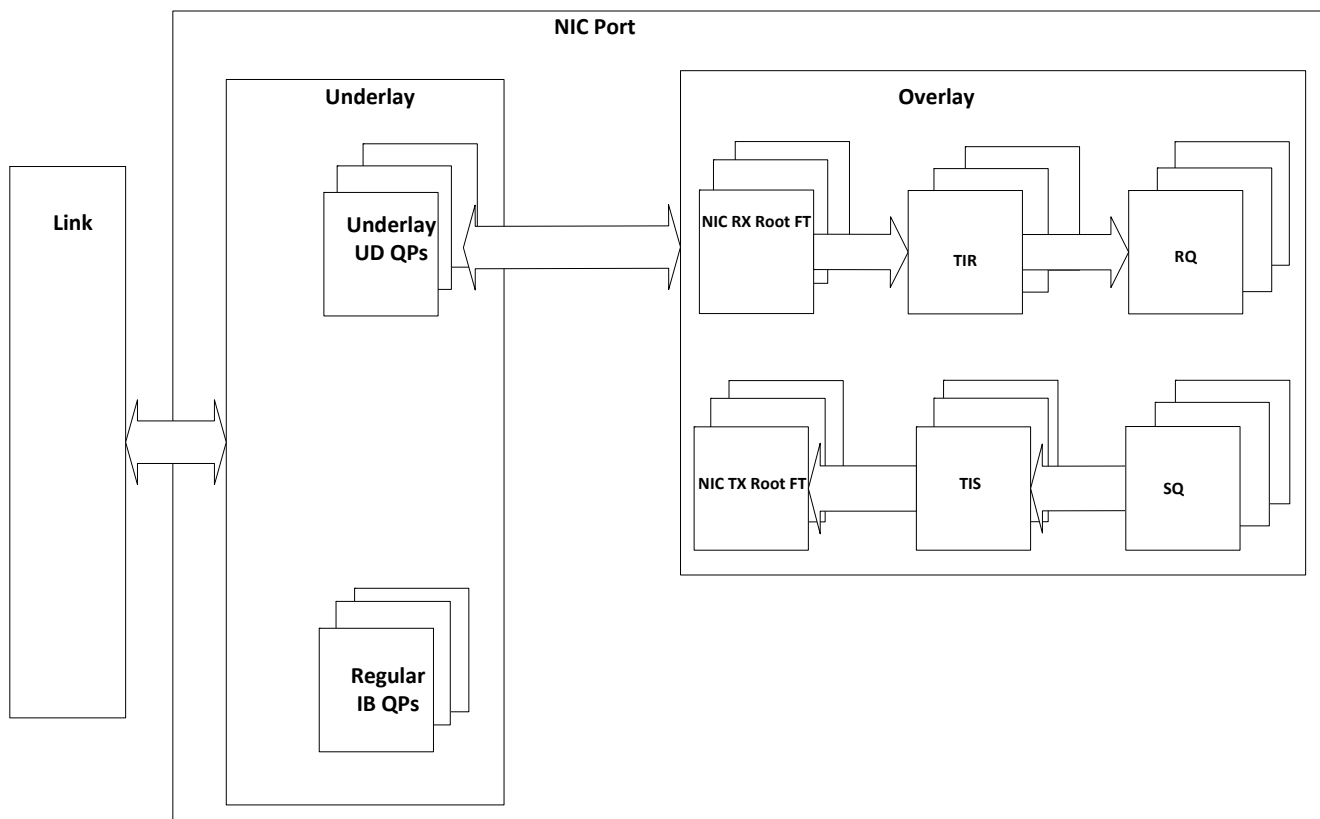
This mode is supported when `HCA_CAP.X_enhanced_offload==1`, where X is the ULP like IPoIB.

QP that works in this mode represents the underlay network- the IB network - and processes only the IB headers of the packet while the “Networking Transport” Objects that “seats above” that QP (Flow Table, TIR, TIS, RQ, SQ, RMP) represents the overlay network and they perform the ULP Stateless Offloads, see Figure TBD.

Note: Underlay QP does not hold any Work Queues and does not deal with packet scattering and gathering. This is done by the “overlay” RQs and SQs that are associated with the QP.

Note: When ULP offloads are enabled Send with Immediate is not supported and extension Address vector is required in send WQEs. TBD: Add it also to Basic section.

Figure 15: ULP Enhanced Mode - Traffic Flow Diagram



4.6.1.2.1 Enhanced ULP Offloads Configuration

In order to configure the enhanced stateless offloads perform the following steps:

- Validate that the enhanced Stateless Offloads are supported using `QUERY_HCA_CAP.ulp_enhanced_offloads` capability.
- Create underlay UD QP with `ulp_stateless_offloads_mode=2`.
- Bring the underlay UD QP to RTS state as done for “regular” UD QP. See [Table 19, “QP Context Structure Layout,”](#) on page 258.
- Create RX NIC overlay Flow Tables. See [Section 8.18, “Flow Table,”](#) on page 414
- Associate the underlay UD QP with RX root Flow Table using `SET_FLOW_TABLE_ROOT` command. See [Section 29.18.4, “SET_FLOW_TABLE_ROOT - Set Flow Table Root,”](#) on page 1995.
- If TX steering is required:
 - Create TX NIC overlay Flow Table.
 - Associate underlay QP with TX root Flow Table using `SET_FLOW_TABLE_ROOT` command. See [Section 29.18.4, “SET_FLOW_TABLE_ROOT - Set Flow Table Root,”](#) on page 1995.

- Allocate and configure the overlay objects (TIR, TIS, RQ, SQ, RMP) that are supposed to perform the ULP enhanced Stateless Offloads.
- Associate overlay TIS with the underlay UD QP by setting TIS.uqpn=Underlay UD QPN. See [Section 8.13, “Transport Interface Send \(TIS\),” on page 381](#).
- Configure the overlay Flow Tables. Note that the valid Ethertype values for IPoIB Flows are IPv4, IPv6, ARP and RARP. Flows with Ethertype match value ARP or RARP will match packets with either Ethertype value (ARP or RARP).

In order to listen to multicast traffic perform the following steps:

- Join the multicast group.
- Attach the underlay QP to the multicast group via ATTACH_TO_MCG.
- Configure overlay Flow Tables via Flow Table commands.

Note: When PKey table changes (indicated by "PKEY changed" event) a ULP QP may use an old PKey. The QP should be recreated if it uses a PKey index pointing to a modified entry. This flow can be skipped if HCA_CAP.ipoib_enhanced_pkey_change is set.

4.6.1.2.2 Enhanced ULP Packet flow

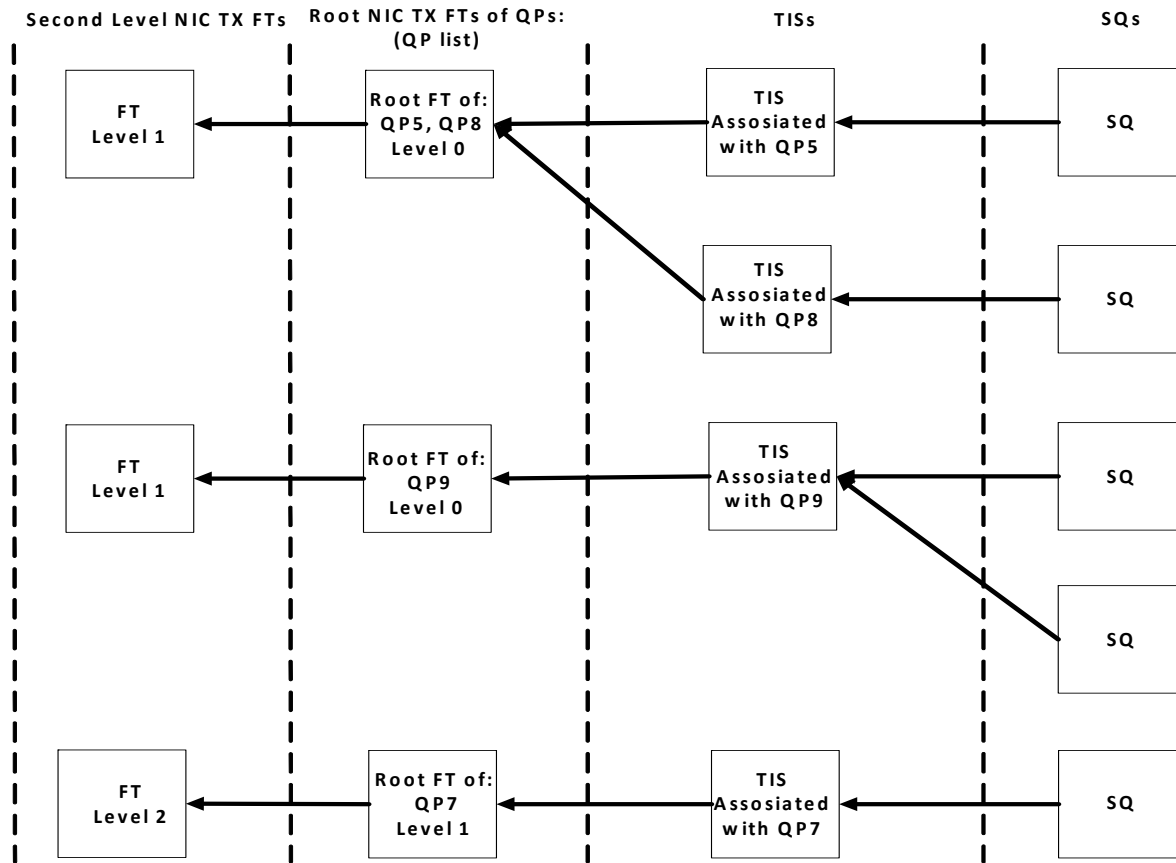
Transmit Side Flow (TX):

- On doorbell, SQ processes the send WQE and generates a packet.
- The packet is delivered to the TIS that is pointed to by the SQ that chooses the underlay UD QP.
- Packet is delivered to chosen QP's Root NIC TX Flow Table.
- NIC TX Flow table rules are applied on the packet's overlay headers (e.g. IP header fields) and if the action is “forward”, the packet is delivered to the underlay QP.

Note: If NIC TX Flow Table is not created, the NIC TX flow table default action is applied - forward the packet to chosen underlay QP.

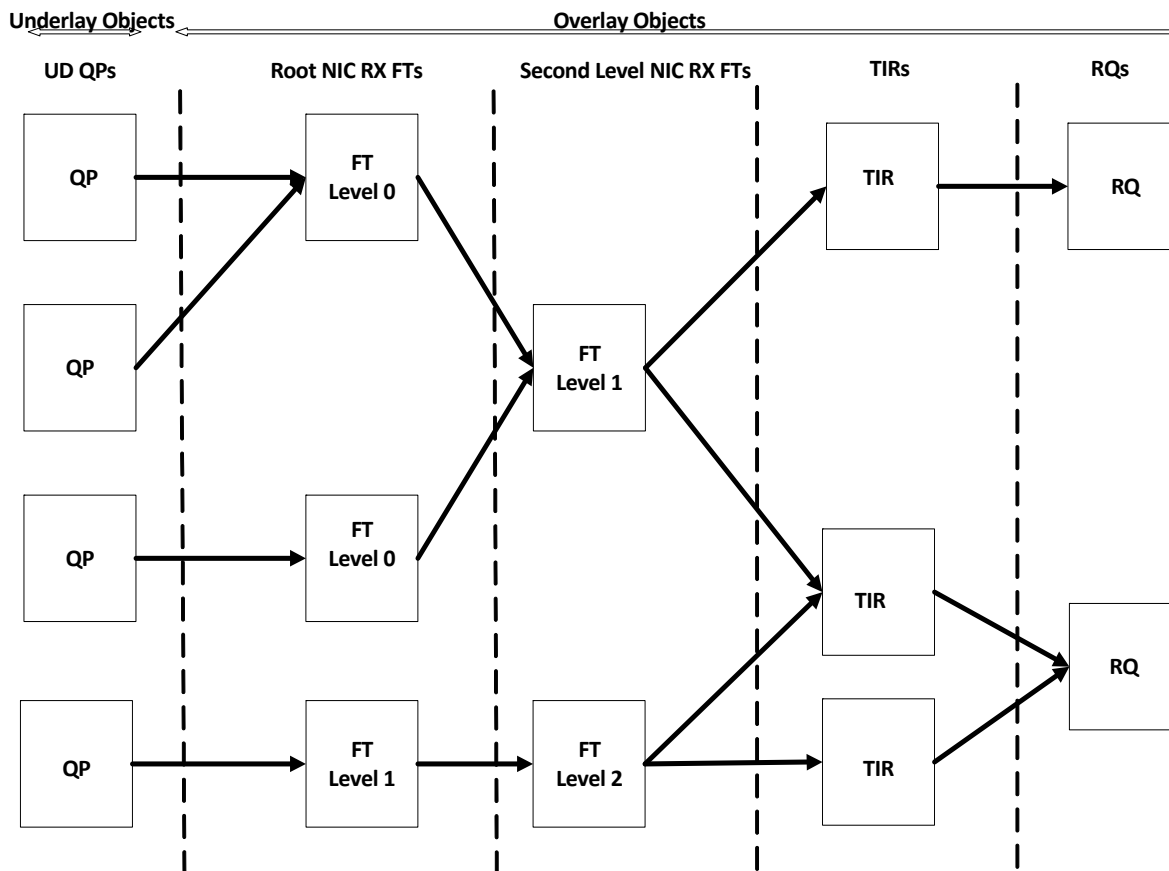
- QP processes the packet, adds the IB (underlay) headers, performs the relevant checks (by IB spec) and sends the packet to the wire.

Figure 16: Transmit Objects Diagram



- The packet is delivered to underlay QP according to IB headers.
- QP processes the packet, strips the IB (i.e. underlay) headers, performs the relevant IB transport checks and delivers the packet to its Root NIC RX Flow Table.
- RX Flow table rules are applied on the packet's overlay headers (e.g. IP header fields) and if the action is forward to TIR, the packet is delivered to TIR/s.
- TIR performs the overlay transport packet processing and forwards the packets to the associated RQ/s.
- RQ process the receive WQE, scatters the packet into memory and generates completion.

Figure 17: Receive Objects Diagram



4.6.1.2.3 Root Flow Table

In order to overlay NIC Receive/Transmit Flow Tables will get incoming/outgoing traffic, they must be associated with underlay UD QP. This is done via `SET_FLOW_TABLE_ROOT` command. See [Section 29.18.4, “SET_FLOW_TABLE_ROOT - Set Flow Table Root,”](#) on page 1995.

Each underlay UD QP can be associated with a single root Flow Table type (one for TX and one for RX), while each root Flow Table can be shared between multiple QPs.

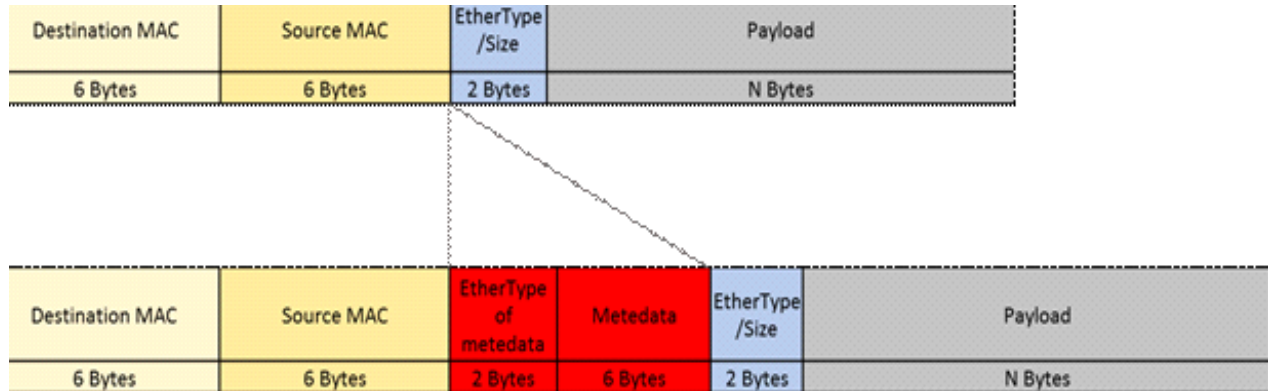
Note: Only QP in RTS state is allowed to be associated with Flow Table, otherwise the behavior is undefined. QP can be moved from RTS state only after it is disconnected from the Flow Tables via `SET_FLOW_TABLE_ROOT.reset_root_to_default` command. See [Section 29.18.4, “SET_FLOW_TABLE_ROOT - Set Flow Table Root,”](#) on page 1995.

4.7 MLNX ETHERTYPE TAGGING

The device has the ability to recognize special tagging depending on special ethertype value reported in `HCA_CAP.mlnx_tag_ethertype` (See Table 1249, “HCA Capabilities Layout,” on page 1378). When the device recognizes this ethertype, it can continue the parsing process and

skip the 6 bytes tag after the ethertype as illustrated in Figure , “MLNX Tagging Packet Format,” on page 215.

MLNX Tagging Packet Format



4.8 Hairpin

This feature is supported when `HCA_CAP.hairpin==1`.

Hairpin provides the ability to send the received traffic from the network back to the network without any CPU processing. The received packet can either be transmitted through the same port it was received from, or from an another port.

As explained below, SW can connect a hairpin RQ with a hairpin SQ. This connection cause all the traffic received on this RQ to be forwarded by the device directly to the SQ.

4.8.1 Hairpin Creation Flow

- CREATE SQ with `SQ.hairpin==1`.
- CREATE RQ with `RQ.hairpin==1`.
- MODIFY SQ to RDY and set `hairpin_peer_vhca` and `hairpin_peer_rq`.
(Internal - FW marks SQ as already connected).
- MODIFY RQ to RDY and set `hairpin_peer_vhca` and `hairpin_peer_sq`.
(Internal - FW marks RQ as already connected, and allows hairpin connectivity only if SQ includes match fields. i.e. *each one point to other).

Note - The only valid fields in WQ of hairpin SQ/RQ are “`log_hairpin_num_packets`” and “`log_hairpin_data_sz`”. The buffers are allocated by the device, and SW should not bypass them.

Note - When one of the hairpin (SQ or RQ) get error, the peer Queue also moved to error and an event is generated.

[Internal Note] To handle error flows, FW has an indication in each context SQ/RQ which allows the connection to be established only once. This indication is cleared when moving 2RST or new creation. This mechanism handles untrusted applications and drivers.

[Internal Note] WQs of hairpin should be allocated as NIM and use strided RQ by FW. SW only provides the size of these WQs.

4.8.2 Destroy Hairpin Flow

- Move RQ 2RST
- Move SQ 2RST
- Destroy RQ
- Destroy SQ

4.9 TLS Offload

TLS data-path offload allows the NIC to accelerate encryption, decryption and authentication of AES-GCM. TLS offload handles data as it goes through the device without storing any of it, but only updating context. If the packet cannot be encrypted/decrypted by the device, a software fall-back handles the packet. For example, rerouting is handled by software. Retransmission or out-of-order packets requires a resynchronization (re-sync) of the device.

4.9.1 Initialization and enable of offload

This feature is supported when `HCA_CAP.tls==1`. detailed capabilities of the feature reported in TLS Capabilities (See [Table 1609, “TLS Capabilities Layout,” on page 1789](#)).

Any key that is used with TLS offload must be created first by `CREATE_GENERAL_OBJ` command for DEK object (See [Section 8.27.14, “DEK Object,” on page 564](#)).

4.9.1.1 Overview

For the transmit-side, a TIS must be created with `tls_en==1` (see [Table 121, “TIS Context Format,” on page 382](#)). The device will create automatically a Transport Static Params Context that can be updated by UMR WQE (see [Section 8.22, “Transport Static Params Context,” on page 525](#)).

After that, this TIS is ready to be used to send WQE with TLS offload.

The TIS created for a TLS is not like a regular TIS that pointed by SQ. It is unique in the sense that it inherits the priority, transport domain, and underlay `qp_number` properties of the TIS context. In essence, these properties of TLS TIS are masked by the properties of the regular TIS pointed by the SQ.

For the receive-side, a TIR must be created with `tls_en==1` (see [Table 116, “TIR Context Format,” on page 374](#)). The device will create automatically a Transport Static Params Context that can be updated by UMR WQE (see [Section 8.22, “Transport Static Params Context,” on page 525](#)).

After that, this TIR is ready to be used to receive WQE with TLS offload.

TLS offload parses and verifies the TLS record headers, deduces their length and tracks their sequence numbers to perform crypto offload. Packets with invalid TLS records (bad magic or record length) will cause an error.

4.9.1.2 Detailed Flow

- Allocate or reuse a TIR/TIS with `tls_en==1` (see [Table 121, “TIS Context Format,” on page 382](#) and [Table 116, “TIR Context Format,” on page 374](#))
- Allocate DEK using DEK Object (see [Section 8.27.14, “DEK Object,” on page 564](#))
- Set `Transport_Static_Params` Context using UMR WQE (see [Section 8.22, “Transport Static Params Context,” on page 525](#)) where:
 - `ctrl.opcode = UMR`
 - `ctrl.size=2`
 - `ctrl.opcodemode = TLS_TIS_STATIC_PARAMS/TLS_TIR_STATIC_PARAMS`
 - `ctrl.ctrl_general_id[31:8] = TIS/TIR`
 - `umr.ctrl.inline = 1/0`
 - `umr.ctrl.Check_free = 0.`
 - `umr.ctrl.klm_offset_en = 0`
 - `umr.ctrl.check_qpn = 0.`
 - `umr.ctrl.skip_mkey_fence = 1/0`
 - `umr.ctrl.klm_octoword_count= 0`
 - `umr.ctrl.bsf_octoword_count= 4`
 - `umr.ctrl.MkeyCtxbitmask= 0`
 - `umr.ctrl.klm_offset[47:16] = 0`
 - `umr.ctrl.Mkey_context = 0`
 - `umr.pointer.mkey = mkey pointer` (mem region which contain the TLS BSF).
 - `umr.pointer.address = pointer to source buffer` (pointer to the TLS BSF)
 - `umr.inline.MTTs/KLMs = size is zero`
 - `umr.inline.BSFs = contains the TLS BSF//` (according to `bsf_octoword_cnt`)
 - `Transport_Static_Params.tls_version`
 - `Transport_Static_Params.const_1 = 1`
 - `Transport_Static_Params.encryption_standard = 1`
 - `Transport_Static_Params.initial_record_number = expected record number`
 - `Transport_Static_Params.resync_tcp_sn = 0`
 - `Transport_Static_Params.gcm_iv = 32 MSB of implicit IV`
 - `Transport_Static_Params.implicit_iv = for TLS1.3 64LSB of implicit IV, reserved for TLS1.2`
 - `Transport_Static_Params.dek_index = index to DEK`

- Set TLS Progress Params Context using SET_PSV WQE(See [Section 8.23, “TLS PROGRESS Params Context,”](#) on page 528) where:
 - next_record_tcp_sn = The TCP sequence number of the next expected record (calculated according to length of current record)
 - hw_resync_tcp_sn= 0
 - record_tracker_state = Start
 - auth_state = no_offload
 - hw_offset_record_number = 0
- For receive operations, set steering action that points to the TLS TIR. Steering action must be set after set_TV and set_TNV operations have completed.
- For send operation, the first send operation must occur only after completion of the context setup.

TLS TIR/TIS contexts can be either deallocated or reused to reduce the number of control operations needed to create a context. It is important to optimize the context creation because it occurs during the TLS handshake.

Detailed Send and Receive flows are described under [Send Flow Description](#) and [Receive operation](#) accordingly.

4.9.2 TLS Send Flow

Data that expects TLS offload must have the expected TCP sequence, otherwise the Send Resync Flow must be used. If the TCP expected sequence is correct, then data is sent by setting the TIS number of the TLS Context in the WQE.CTRL.ctrl_general_id[31:8] (also for LSO_SEND and Dump WQEs).

All headers including TCP and trailers (anything which is not TCP payload) must be included inline in the WQE, because the TLS engine will process all the data in the memory gather entry.

Note: In TLS Tx, software must use the WQE fence bit on both the first rebuild step, and also on the first wqe with data sent after the context has changed. This will guarantee that the data is sent after the context has been successfully updated in hardware, and that the context modification doesn't interfere with existing traffic.

4.9.2.1 Out of Order - Resync Flow

If the TCP sequence doesn't match the context's expected TCP sequence. Then we use the send resync flow. In this flow, KTLS provides the data of the record of the TCP sequence that is being transmitted.

The context is rebuilt in two stages:

- **Rebuild step1:** Set the TLS context according to the TLS record sequence number of the record being sent. This packet could be a retransmission so we have to update the device context.

- Optional – set Transport Static Params Context– this step is needed only if the initial_record_number has incremented (use fence)
- Set TLS Progress Params – record_number_HW(SW should set small fence).
- Set TLS Progress Params – update next_record_tcp_sn and record_tracker=start (SW should set small fence).
-
- **Rebuild step2:** Send the data of the TLS record up to the packet that need to transmit using a dummy WQE with the TLS context in the wqe (use fence)
- Send the original packet as any other TLS packet that expects the buffer.

4.9.3 TLS Receive Flow

TLS offload status is reported in special fields in CQE. CQE processing is as follows:

- According to CQE.tls_offload:
 - For partial offload. Software must know that HW encountered an error while processing the TLS records in the packet. As a result, only part of the packet is decrypted. The driver must re-encrypt the entire packet and let the TLS layer figure out what went wrong.
- If CQE.tls_offload==resync is set, then resync is needed mark the socket to call the driver for the resync flow once the TLS record is processed in the TLS layer.

4.9.3.1 Resync Flow

Resync flow occurs when packets have been lost and the device lost track of TLS records. The device attempts to resync by tracking TLS records, and sends a resync request to software. The TLS Progress Params Context holds the TCP-SN of the record where the device began tracking records and counting them. Software will acknowledge the TCP-SN if it matches a legal record by setting the Transport Static Params Context context. Detailed flow follows:

- A CQE.tls_offload==resync asserted is received
- Get TLS Progress Params Context by Get_PSV Wqe (See Section 8.23, “TLS PROGRESS Params Context,” on page 528)
- Keep the hw_resync_tcp_sn value in the socket.
- When all missing data has been received, calculate the corresponding record number (= new initial record number) for the saved hw_resync_tcp_sn value.
- Check the state in the TLS Progress Params context and verify it is in tracking state and either no-offload state. If the state doesn’t match, then exit without updating the Transport Static Params Context.
- Set Transport Static Params (by UMR WQE) – override the initial_record_number with the new value. Override the tcp_sn_resync field with the value saved from the get operation.

- No fencing required.

The device decrypts and authenticates using the initial `record_number + hw_offset_record_number`. At an event of packet drop, when the device attempts to resync, the `hw_offset_record_number` is reset, and software must write to the initial `_record_number` the number of records to be added.

4.9.4 TLS Teardown

The TLS stack should request to release a TLS context when the all references to the TCP socket have been released. As a result, we know that there are no packets in flight that software expects to handle.

For the receive-side, the steering entry pointing to the TIR must be removed before reusing/destroying the TIR.

The keys must be destroyed after the steering entry has been changed. In case of send flow, Keys can only be removed after completion of send WQE.

4.10 NVMeoTCP Offload

This feature is supported when `HCA_CAP.nvmeotcp_en==1`.

4.10.1 Overview

NVME over fabrics and NVME over TCP specifications define how to map the semantics of the NVME storage protocol to the TCP network protocol. Traditionally, NVME over TCP establishes a TCP socket per NVME queue. In turn, NVME requests to read/write data to/from buffers are sent using *NVME command capsule* messages over TCP.

NVME initiators must place data into a pre-specified user buffers. However, TCP implementations provide data that is scattered to the NIC's receive ring, then this data is re-ordered in memory by TCP. As a result, NVME over TCP is forced to copy data out of TCP buffers into its buffer. This data copy incurs a heavy price on system performance.

NVME over TCP data-path offload allows the NIC to accelerate this data placement for the NVME over TCP storage protocol without fully offloading TCP logic to hardware. This offload handles data as it is going through the device without storing any data, but only updating contexts. If the packet data cannot be placed by the device in a predefined location, then a software fallback handles the packet. Successful offload splits a packet into two parts:

- NVME data is placed directly in the NVME buffer.
- TCP and NVME headers (and metadata) are placed in the NIC's receive ring.

The device driver is responsible of constructing a packet that reflects the original packet correctly to higher layers. For example, a driver could construct a packet descriptor which points to the NVME buffer for some of the payload. Then, the NVME layer can skip the copy of this area of the packet payload as the source address is equal to the destination address for this memory copy operation.

4.10.2 CRC offload overview

Additionally, NVME PDU CRC calculation (transmit) and verification (receive) is performed by hardware. On transmit, NVME data is sent with the CRC field empty, hardware will calculate the CRC over the data and fill the empty field. On receive, the CRC of NVME data is checked and an indication of verification success/failure is passed in the CQE.

4.10.3 Context Setup Description

NVMEoTCP offload begins with the user configuring an NVMEoTCP block device (`/dev/nvme1`). The block device creates a TCP socket per queue. In turn, an NVMEoTCP handshake is performed over this socket. Finally, when the TCP socket is ready to receive/send IO commands the device driver is called to configure offload for this TCP socket which begins offload configuration by the device driver.

The driver first creates a TIR object with `nvmeotcp_en`, the CCID table mkey and `nvmeotcp` queue depth. Thereafter, the Static params context is filled and control is returned to the NVME layer. Finally, it sets up a steering entry according to the TCP flow's 5-tuple to point to the TIR object (If steering is set using a wqe, then a fence is sufficient)

4.10.3.1 Detailed flow

- Allocate umem with a KLM entry for each NVMEoTCP command capsule. The maximum number of KLM entries must be $\leq 2^{\text{HCA\_CAP.log\_max\_queue\_depth}}$
- Allocate a NVMEoTCP TIR:
 - Enabled `nvmeotcp_en` (set `nvmeotcp_en` bit in the TIR)
 - Provide the `umem_id` created in the previous step and the size of the `umem_id`
 - The TIR must point to a RQ which point to a CQ that supports 128B CQEs
- Set Static params using a UMR WQE:
 - `ctrl.opcode = UMR`
 - `ctrl.size=2`
 - `ctrl.opcodemode = TRANSPORT_OFFLOAD_STATIC_PARAMS`
 - `ctrl.ctrl_general_id[31:8] = TIS/TIR number`
 - `umr.ctrl.inline = 1/0`
 - `umr.ctrl.Check_free = 0.`
 - `umr.ctrl.klm_offset_en = 0`
 - `umr.ctrl.check_qpn = 0.`
 - `umr.ctrl.skip_mkey_fence = 1/0`
 - `umr.ctrl.klm_octoword_count= 0`
 - `umr.ctrl.bsf_octoword_count= 4`
 - `umr.ctrl.MkeyCtxbitmask= 0`

- umr.ctrl.klm_offset[47:16] = 0
- umr.ctrl.Mkey_context = 0
- umr.pointer.mkey = mkey pointer (mem region which contain the TLS BSF).
- umr.pointer.address = pointer to source buffer (pointer to the TLS BSF)
- umr.inline.MTTs/KLMs = size is zero
- umr.inline.BSFs = contains the TLS BSF// (according to bsf_octword_cnt)
- NVMEOTCP_Static_Params.BSF_P_TYPE = 2 (could be const/reserved)
- NVMEOTCP_Static_Params.encryption_standard = 0 (could be const)
- NVMEOTCP_Static_Params.PDA = padding alignment
- NVMEOTCP_Static_Params.ti = Target (1) or initiator (0)
- NVMEOTCP_Static_Params.NVME_resync_tcp_sn = TCP sequence of the next expected byte.
- NVMEOTCP_Static_Params.ddgst_en = 1 if ddgst is enabled.
- NVMEOTCP_Static_Params.ddp_en = 1 if direct data placement on receive is enabled.
- Set Progress Params using a SET_PSV WQE:
 - ctrl.opcode = SET_PSV
 - ctrl.size=2
 - ctrl.opcodemode = NVMEOTCP_TIR_OFFLOAD_PROGRESS_PARAMS
 - Set the initial expected sequence number according to the next expected NVME Command Capsule TCP sequence.

4.10.3.2 Receive operation

NVMEoTCP offload utilizes new CQE bits to convey info about the result of offload. The CQE information describes the location where the NVME data is scattered by hardware offload, e.g. in the NVME buffer. For details see CQE extensions.

CQE processing is as follows:

- According to CQE.nvmeotcp_offload fill the packet descriptor fields such that all data of the packet is accounted for in this descriptor.

TBD - Boris add more details about all cqe fields

4.10.3.3 Resync flow

Resync flow occurs when packets have been lost and the device lost track of NVMEoTCP PDUs. The device attempts to resync by tracking NVMEoTCP PDUs, and sends a resync request to software. The CQE of the record where the device began tracking records and counting them. Software will acknowledge the TCP-SN if it matches a legal NVME capsule by setting the Static params context. Detailed flow follows:

A CQE with “resync_bit” asserted is received:

- Read the `hw_resync_tcp_sn` value from the CQE and place it on the NVMeoTCP queue.
- When all missing data has been received, check that the received TCP sequence number (`=CQE.hw_resync_tcp_sn`) is truly an NVME PDU header.
- Set NVMeoTCP static params – write `hw_resync_tcp_sn` to the `tcp_sn_resync` of the static params. Set the WQE small fence bit on this WQE.

4.11 IPSEC Offload

TBD

4.12 DPP Offload - Direct Packet Placement Offload

This feature is supported when `HCA_CAP.dpp=1`. Detailed capabilities are reported in Table 1613, “DPP Capabilities Layout,” on page 1792.

Direct Packet Placement (DPP) is a receive side transport service, in which the Ethernet packets are scattered to the memory, according to a packet sequence number (PSN) carried by the packet, and not by their arrival order. Assuming the flow is carrying a big application segment, DPP will cause the device to assemble the application buffer, with no additional copies.

Per flow, SW allocates and registers a big enough buffer. This buffer is described using MKEY.

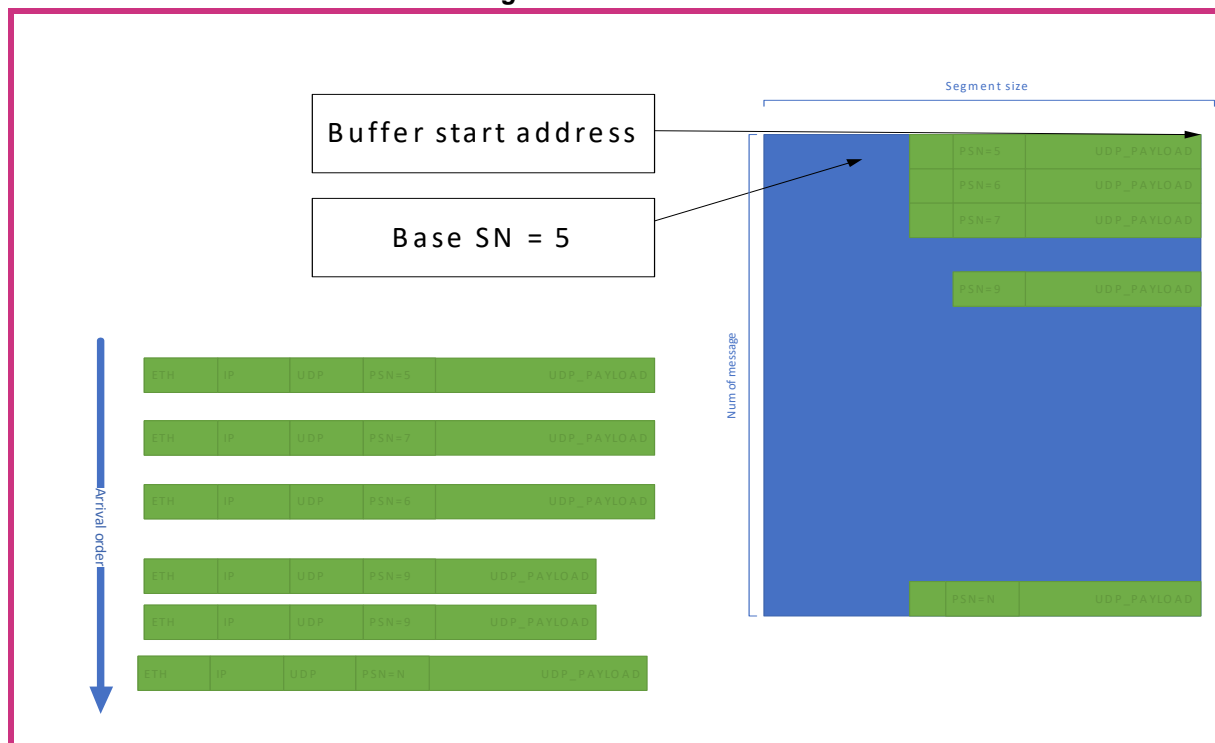
The packet scatter address will be:

$\text{Mkey\_Start\_Address} + ((\text{PSN} - \text{BaseSn}) \% (\text{number of packets in buffer})) * \text{dpp\_segment size}$

- Mkey_Start address – indicated by the MKEY
- PSN – the sequence number within the packet, unique for every packet
- BaseSn – the sequence number that belongs to the first segment in the buffer
- Number of packets in the buffer = buffer size / segment size
- Segment size – the length between the scattering address of one PSN to the next

4.12.1 Scattering Overview

Figure 18: DPP Scatter to Buffer



A flow, is transmitting packet, carrying a PSN which indicate the packet order within a flow, and also implicitly hinting with data is carried by that packet. The (good) example is Video flow, and but it's not the only case.

One of the purposes of this transport service is to offload real time protocols (RTP) and accelerate software application, by assembling the application buffer directly from the NIC and by the hardware.

Specifically, direct packet placement can be used to offload the following protocols, for example:

- SMPTE 2022 -5/6/7
- VSF TR-03/IETF RFC 4175
- intoPIX TICO
- Sony IP Live RDD 34 and NMI
- NewTek
- GigE vision GVSP

4.12.2 DPP Offload Creation

To enable DPP offload, SW should create a special RQ by CREATE_RQ command, and set:

- mem_rq_type= DPP

- `Dpp_wire_protocol`
- `Dpp_mkey`
- `Dpp_buffer_size`
- `Dpp_segment_size`

WQ and all other fields in RQ are reserved.

A flow is determined by a Flow Table Entry rule that points to a TIR. This TIR points to RQ with `mem_rq_type = DPP`.

For each received packet, a CQE will be generated. In addition to the regular CQE fields, it also includes:

- The packet sequence number

The total number of bytes scattered.

5 PCI Interface

The device accesses the host processor through a 16x PCI Express 3.0 bus, capable of consuming/driving data at a rate of 16Gbyte/sec full-duplex.

The device is capable of presenting up to 16 physical functions and 256 virtual functions. **Each physical function can be assigned to either (or both) network ports and virtual functions can be associated with any physical function to the full extent of PCIe spec flexibility.**

Each physical and virtual function exposes PCI type0 configuration header.

Each function (be it physical or virtual) exposes a single BAR called HCA BAR - a prefetchable BAR used to post control data path commands. Both the initialization segment and the UAR pages are implemented on this BAR. This is Bar0 in the PCI header.

[TBD - what is really in HCA BAR](#)

[TBD - HCA BAR diagram](#)

Format of the initialization segment is shown in “Initialization Segment” on page 234. Format of the UAR pages is shown in “User Access Region (UAR)” on page 249.

5.1 PCIe Compliance

The device is PCI Express 3.0 compliant. It supports the following features of the PCI Express 3.0:

- Optimized Buffer Flush/Fill (OBFF)
- Latency Tolerance Reporting (LTR)
- Re-sizable BARs
- L1 Active State Power Management (ASPM)
 - ASPM Optionality
- Advanced Error Reporting
- Alternative Routing-ID Interpretation (ARI)
- Single Root I/O Virtualization (SR-IOV)
- ID-Based Ordering (IDO)
- Extended Tag
- TLP Processing Hints (TPH)
 - Steering Tags are not supported

The device also supports:

- 8.0 GT/s Receiver Impedance
- Precision Time Measurement (PTM)
- Readiness Notifications (RN)

The following features are supported in memory region granularity:

- No-snoop
- Relaxed ordering

- TPH - TLP processing hints

[See Table 362, “MKey Context Format,” on page 629.](#)

[TBD - the following paragraph is wrong. this is in consistent with other diagrams.](#)

[TBD - multi-host managementTBD - rewrite the paragraph below](#)

[TBD - does a non trusted driver run on the VF or the VM](#)

5.2 Capabilities Reporting

PCI Express® Base Specification defines a set of capabilities which should be reported when made available by the device. These capabilities are reported through a special structure of linked-list items. The first capability is pointed in offset 0x34 in the configuration space, and the last capability is indicated by setting its “next capability” pointer to 0x00.

The capabilities are defined in Appendix H of the PCI Express® Base Specification. PCIe additional capabilities (added on top of PCI) are described in Section 7.5 of the same specification.

The capabilities reported by the device are listed in Table 8.

Table 8 - PCI Capabilities Summary

Name	ID Value	Description
Configuration header		Type 0 Configuration header
Power	0x1	PCI Power Management Interface Defined by <i>PCI Power management interface specification</i> .
VPD	0x3	Vital Product Data (VPD) Defined in section 6.4 in Appendix I of <i>PCI Local base specification, Revision 3.0</i>
PCI Express	0x10	PCI Express Defined in section 7.8 of <i>PCI Express Base Specification, Revision 3.0</i>
MSI-X	0x11	MSI extension support Defined in section 6.8 of <i>PCI Local base specification, Revision 3.0</i>
MSI	0x5	Message Signaled Interrupts (MSI) Defined in section 6.8 of <i>PCI Local base specification, Revision 3.0</i>
Vendor Specific	0x9	See “VSC - Vendor Specific Capability”

5.3 VSC - Vendor Specific Capability

The PCI Vendor Specific Capability is defined in Table 9, “Vendor Specific Capability Layout”. It is used to activate a Gateway, interfacing with the device. The Gateway is used to read or write

device configurations, which are organized in different domains. A configuration access may result in multiple actions, denoted as a command.

Note: This is a Vendor specific capability (VSC) and not a Vendor Specific Extended Capability (VSEC) as it is located in the PCI PCI configuration space and not in the extended PCI Express configuration space. However, the terms VSC and VSEC will be used interchangeably throughout this document.

Table 9 - Vendor Specific Capability Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																
type								length								next_pointer								capability_id								00h																
status		size vld	syn- drome_code													space															04h																	
																																																08h
																semaphore																																0Ch
																syndrome flag	address																														10h	
																																															data	

Table 10 - Vendor Specific Capability Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	type	Describes the capability type. Set to 0x0.	RO
	23:16	length	Provides the number of bytes in the capability structure (including the Capability ID and Next pointer bytes).	RO
	15:8	next_pointer	Pointer to the next item in the capabilities list. Value 0x0 indicates the final item in the list.	RO
	7:0	capability_id	Vendor specific capability. Set to 0x9.	RO

Table 10 - Vendor Specific Capability Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:29	status	0x0: Space is not supported 0x1: Space is supported Else - Reserved	RO
	28	size_vld	When set, the maximal address of the space is indicated in address field, after writing the space field.	RO
	27:24	syndrome_code	Error type when accessing an address using the Gateway. Valid when a Read/Write operation is complete and the <i>syndrome</i> field is set. The <i>syndrome_code</i> is cleared upon any write of the Gateway owner to the VSC. [Unimplemented, deprecated for syndrome info in data field] 0x0: UNSPECIFIED 0x1: READ_ONLY_SPACE 0x2: READ_ONLY_ADDRESS 0x3: ADDRESS_OUT_OF_RANGE 0x4: RESTRICTED_ADDRESS 0x5: BUSY other values are reserved.	RO
	15:0	space	Selects which domain to access: 0x0: Reserved so that un-initialized space will go to NULL. 0x1: ICMD 0x2: CRSPACE 0x3: All ICMD 0x4: NODNIC initialization segment 0x5: Expansion ROM 0x6: ND CRSPACE 0x7: SCAN_CRSPACE 0xA: Global_Semaphore 0xF: MAC Other values are reserved	RW
08h	31:0	counter	Self- Incrementing counter. Possible values are 0x0001 – 0xFFFF. Never reads as 0x0.	RO
0Ch	31:0	semaphore	On writes – request ownership by writing the previously read counter value. On reads – 0x0 – GW is free, else GW is locked.	RW
10h	31	flag		RW
	30	syndrome	When set, the address access has failed. Valid only when the access is complete (according to flag field). An extended fail information can be seen on data field.	RW
	29:0	address	Address pointer to the selected address space. After writing <i>space</i> field, if <i>size_vld</i> field is set, this field holds the maximal valid address of the space	RW
14h	31:0	data	Data word to be written to/read from the pointed location. If a Gateway transaction fails (according to syndrome field), this field holds additional failure information.	RW

5.3.1 Configuration Domains

The following configuration domains may be accessed using the Gateway.

5.3.1.1 ICMD Space

ICMD interface is an interface method allowing a SW tool to send commands to the device. The interface is intended to allow the tools to operate in parallel and independently from the host software.

5.3.1.2 CRSPACE Space

This space exposes the on-chip registers and resources of the device.

5.3.1.3 All ICMD Space

This space is identical to the map of ICMD space only that it allows the use of all the defined ICMDs (including commands not exposed to external driver developers).

5.3.1.4 Expansion ROM

This Read-Only space is an alternative to the standard PCI 'expansion ROM BAR' (also called BIOS option ROM BAR). Expansion ROM bootstrap code can use this space in order to copy the whole expansion ROM to RAM. This method is useful for 32bit loaders when the 64bit expansion ROM bar is mapped to an address above 4GB.

Note: The NIC expansion ROM (Flexboot) has support for iSCSI, PXE, HTTP boot, configuration and other services. The expansion ROM may not be present in all models.

5.3.1.5 Nondestructive (ND) CRSPACE Space

This Read-Only space exposes the on-chip registers and resources of the device, for Read access, in a nondestructive manner. Accesses through this space will not modify the device state.

5.3.1.6 Scan CRSPACE Space

This Read-Only space exposes the on-chip registers and resources of the device for Read access, in a nondestructive manner. Accesses through this space will not modify the device state.

Note: This space is optimized for scanning the entire space. When completing a “Gateway Read”, the **address** field will hold the next valid header in the space. Using this address for the next Read will avoid accessing redundant addresses, making the scan faster. When accessing the last valid address of the space (or beyond), the address field will hold the same value as requested, indicating there are no more valid addresses beyond it.

5.3.1.7 NODNIC Initialization Segment

This space exposes the Initialization segment. The Gateway allows accessing this segment using PCI configuration cycles, instead of accessing the HCA BAR. NODNIC configuration registers are only accessible through this Gateway. See [Chapter 12, “No-DRAM NIC” on page 813](#).

5.3.1.8 Global Semaphores Space

This space exposes an array of 32bit semaphores. Each Semaphore is defined to synchronize access to a specific resource or service.

When a process on a PF needs to lock a semaphore, it writes a non-zero Unique-ID value to the semaphore entry. If the semaphore is successfully locked for this specific process, then reading the semaphore value returns the same Unique-ID number. To release a semaphore, the owning process has to write a value of 0x00 to the semaphore.

The entries are shared between all PFs in the device(TBD - including PFs assigned to multiple hosts) and can be used to coordinate between them. The Unique-ID should be read from the counter field in the VSC Gateway, or the same value used to lock that Gateway

Unique-ID 0xFFFFFFFF is reserved, and indicates the Semaphore is not supported.

[Internal] The FW stores for each of these entries the PF#, host# and process-ID, which requested locking of the specific resource, and a flag indicating of the resource is free or locked. By storing these parameters, FW can reset resources which were locked to PF which becomes non-responsive. Internally, the FW will check the semaphore value to determine if to accept the semaphore ownership request. If the semaphore value is 0x0 (semaphore is free) the FW will accept the request. If it is already locked (not 0x0) then the FW will refuse the request.

Table 11 describes the mapped Semaphores.

Table 11 - Global Semaphore Array

Index	Address	Resource	Comments
0	0x0	Tools ICMD	Shared among all PFs
1	0x10	Flash Programming	Shared among all PFs
2	0x20	Software Reset (SW Reset)	Shared among all PFs
3	0x30	Reserved	Shared among all PFs
4	0x40	Reserved	Shared among all PFs
5	0x50	Reserved	Shared among all PFs
6	0x60	Reserved	Shared among all PFs
7	0x70	Reserved	Shared among all PFs

5.3.1.9 MAC Space

This space exposes the MAC address assigned to the function. MAC Space does not behave as other spaces, and does expose a Gateway. The format of the VSC for MAC space is given by Table 12, "Vendor Specific Capability, MAC Space Layout".

Table 12 - Vendor Specific Capability, MAC Space Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type								length								next_pointer								capability_id								00h
status																space															04h	
counter																																08h
semaphore																																0Ch
																mac[47:32]																10h
mac[31:0]																																14h

Table 13 - Vendor Specific Capability, MAC Space Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	type	Describes the capability type. Set to 0x0.	RO
	23:16	length	Provides the number of bytes in the capability structure (including the Capability ID and Next pointer bytes).	RO
	15:8	next_pointer	Pointer to the next item in the capabilities list. Value 0x0 indicates the final item in the list.	RO
	7:0	capability_id	Vendor specific capability. Set to 0x9.	RO
04h	31:29	status	0x0: Space is not supported 0x1: Space is supported Else - Reserved	RO
	15:0	space	Selects which domain to access: 0x0: Reserved so that un-initialized space will go to NULL 0x1: ICMD 0x2: CRSPACE 0x3: All ICMD 0x4: NODNIC initialization segment 0x5: Expansion ROM 0x6: ND CRSPACE 0xA: Global_Semaphore 0xF: MAC Other values are reserved	RW
08h	31:0	counter	Self- Incrementing counter. Possible values are 0x0001 – 0xFFFF. Never reads as 0x0.	RO
0Ch	31:0	semaphore	On writes – request ownership by writing the previously read counter value. On reads – 0x0 – GW is free, else GW is locked.	RW
10h	15:0	mac[47:32]	MSB of unique the MAC assigned for the function	RW
14h	31:0	mac[31:0]	LSB of unique the MAC assigned for the function	RW

5.3.2 Gateway Ownership

A requesting task, must gain the Gateway ownership before it is allowed to use it. To gain the ownership, a task should go through the following steps:

1. Check whether the Gateway is free or busy, by reading the **semaphore** field. If it reads as 0x0 the Gateway is free, otherwise it is busy. If it is busy, the task should wait a random time between 0 to 10msec before repeating this step.
2. Read a value from the **counter** field and write it into the **semaphore** field.
3. Read back the **semaphore** field. If the value returned is the same value the task wrote on step 2, then the task has ownership of the Gateway. If a different value is returned, go back to step 1.

When the Gateway operations are done, the task should relinquish its ownership by writing a value 0x0 to the **semaphore** field.

Each PF exposing this capability has its own gateway, therefore, locking one gateway's semaphore will not interfere with another PF or another host.

Upon warm reboot, the gateway is reset and the semaphore is returned to a free state.

5.3.3 Gateway Read

A Read operation using the Gateway is performed as follows:

1. Write the required configuration domain in **space** field.
2. Read the space parameters (may be done once for all accesses for the space):
 - status - indicate the space is supported
 - **size_vld** - indicate if the space maximal address is available
 - **address** - indicate the maximal valid address of the space
3. Write the required configuration address in **address** field, and clear the **flag** and **syndrome** field to 0x0.
4. Once the Read operation completes the **flag** field will be set to 0x1 by the device. **The syndrome field will indicate the success of the access, and the syndrome_code will provide additional information in case the access failed.**
5. When the flag field is set, read the **data** field to acquire the requested configuration.

Note: Writing to **address** field or **data** field while the **flag** field is not set may cause an unpredictable result for the Read operation.

5.3.4 Gateway Write

A Write operation using the Gateway is performed as follows:

1. Write the required configuration domain in **space** field.
2. Read the space parameters (may be done once for all accesses for the space):
 - status - indicate the space is supported
 - **size_vld** - indicate if the space maximal address is available

- **address** - indicate the maximal valid address of the space
- Write the required configuration data in **data** field.
 - Write the required configuration address in **address** field, and set the **flag** field to 0x1 and **syndrome field to 0x0**.
 - Once the Write operation completes the **flag** field will be cleared to 0x0 by the device. **The syndrome field will indicate the success of the access, and the syndrome_code will provide additional information in case the access failed.**

Note: Writing to **address** field or **data** field while the **flag** field is not cleared may cause an unpredictable result for the Write operation.

5.4 Initialization Segment

The initialization segment is located at offset 0 of HCA BAR. The device uses this segment in the initialization flow (as described in [Chapter 7, “Initialization and Teardown” on page 243](#)), and to perform command doorbell as further described in [Chapter 29, “Command Reference” on page 1647](#).

Table 14 - Initialization Segment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fw_rev_minor																fw_rev_major																00h
cmd_interface_rev																fw_rev_subminor																04h
																																08h-0Ch
cmdq_phy_addr[63:32]																																10h
cmdq_phy_addr[31:12]																					nic_in- terface	log_cmdq_ size		log_cmd- q_stride		14h						
command_doorbell_vector																																18h
																																1Ch-1F4h
																														traf- fic_st ate		1F8h
initializing	nic_interface_sup- ported				embedded_cpu					<u>initializ- ing_state</u>																		1FCh				
	health_buffer (See Table 813, “Health Buffer Layout,” on page 1049)																															

Table 14 - Initialization Segment

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
no_dram_nic_offset																																240h
																																244h
																															capl_invalidate	248
																																24Ch-FFCh
internal_timer																																1000h-1004h
																																1008h
																															clear_int	100Ch
health_syndrome								health_counter																								1010h
																																1014h-2008h
																												clear_intx				200Ch
																																2010h-4008h

Table 15 - Initialization Segment Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	fw_rev_minor	Firmware Revision - Minor	RO
	15:0	fw_rev_major	Firmware Revision - Major	RO

Table 15 - Initialization Segment Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:16	cmd_interface_rev	Command Interface Interpreter Revision ID This number is bumped up every time a non-backward-compatible change is done for the command interface.	RO
	15:0	fw_rev_subminor	Firmware Sub-minor version (Patch level)	RO
10h	31:0	cmdq_phy_addr[63:32]	Physical address of the command queue record. This field should not be modified after INIT_HCA until TEARDOWN_HCA.	RW
14h	31:12	cmdq_phy_addr[31:12]	Physical address of the command queue record. This field should not be modified after INIT_HCA until TEARDOWN_HCA.	RW
	10:8	nic_interface	NIC interface mode 0x0: full_driver 0x1: disabled 0x2: no_dram_nic 0x7: reset - When a PF does the reset, the whole device, except for the PCI I/F, is reset. This is a device-wide operation affecting all functions. When a VF does the reset, only the VF itself is reset.	RW
	7:4	log_cmdq_size	Log number of cmdqs available Currently - 32 commands	RO
	3:0	log_cmdq_stride	Stride between start of each cmdq Current value - 64B	RO
18h	31:0	command_doorbell_vector	Bit per command in the cmdq. When the bit is set, when writing this vector to the device, the command is moved to HW ownership (HW need to execute the command). bit 0 is related to the command in offset 0 in the command queue etc. The valid bits in this vector are [number of commands-1..0] When driver is in No DRAM NIC mode, this field must not be written.	WO
1F8	1:0	traffic_state	Control the traffic on the port. supported only when HCA_CAP.traffic_fast_control==1. 0x1: DISABLE_TRAFFIC. 0x2: ENABLE_TRAFFIC.	RW

Table 15 - Initialization Segment Field Descriptions

Offset	Bits	Name	Description	Access
1FC	31	initializing	1 - device still in initializing state. 0 - device is ready to receive commands.	RO
	30:24	nic_interface_supported	Bitmask indicating which <i>nic_interface</i> modes are supported Bit 0: full_driver Bit 1: disabled Bit 2: no_dram_nic Bit 6: reset	RO
	23	embedded_cpu	If set, indicates that vhca is running on the embedded CPU and not on the host. See TBD	
	19:16	<u>initializing_state</u>	<u>Indicated the level of initializing.</u> <u>Valid only when initilizing==1.</u> <u>0x0: NOT_ALLOWED</u> <u>0x1: WAITING_PERMETION</u> <u>0x2: WAITING_RESOURCES.</u> <u>0x3: ABORT - Ignore function.</u>	<u>RO</u>
200h-23Ch	512	health_buffer (See Table 813, “Health Buffer Layout,” on page 1049)	Advanced debug information. See See Section 27.2, “Health Buffer,” on page 1049 below.	RW
240h	31:0	no_dram_nic_offset	The offset in bytes, inside the initialization segment, where the NODNIC registers can be found. See Table 660, “NODNIC Configuration Registers Layout,” on page 815. Note: NODNIC registers can only be accessed through “VSC - Vendor Specific Capability”, and not through the BAR.	RO
248h	0	capi_invalidate	If set, invalidate the CAPI translation cache. Once this bit is set the HCA invalidates the CAPI translation cache and does the required fencing to guarantee that translations obtain prior to the setting of this bit will not be used after it is cleared. Supported only when HCA_CAP.capi_invalidate==1	WO
1000h-1004h	64	internal_timer	Current internal timer value. See Section 8.19.12, “CQE Timestamping,” on page 471	RO
100Ch	0	clear_int	Write 0x1 to clear legacy interrupt. See Section 12.7, “Completions and Interrupts,” on page 826.	WO

Table 15 - Initialization Segment Field Descriptions

Offset	Bits	Name	Description	Access
1010h	31:24	health_syndrome	Syndrome 0x1: FW_INTERNAL_ERR - assert triggered in FW code 0x7: DEAD_IRISC - Irisc not responding 0x8: HW_FATAL_ERR 0x9: FW_CRC_ERR 0xA: ICM_FETCH_PCI_ERR 0xB: ICM_PAGE_ERR 0xC: ASYNCHRONOUS_EQ_BUF_OVERRUN 0xD: EQ_IN_ERR 0xE: EQ_INV 0xF: FFSER_ERR 0x10: HIGH_TEMP_ERR	RO
	23:0	health_counter	See Section 27.2, “Health Buffer,” on page 1049	RO
200Ch	3:0	clear_intx	[Internal] this field is deprecated writing 1 to bit[N] will clear interrupt number N	WO

5.5 Data Interface

5.5.1 PCIe Attributes

The device supports flexible configuration of PCIe cycle attributes. The PCIe cycles' attributes are configured for each Memory Key Context at its creation (see “MKey Context” on page 567). Those attributes are passed to the PCIe interface unit and to the PCIe packets. Note that for indirect address translation where multiple context accesses are involved during the address translation process, the attributes passed to the PCIe interface unit are bitwise OR of all attributes of MKCs involved in address translation.

6 Memory Resources and Utilization

To control its operation, the HCA uses various data structures (contexts) residing in system memory. The HCA requests memory pages from software and then uses those pages to manage these data structures.

The amount of memory to be used depends on the capabilities that the HCA is required to support. For example, the number of concurrently open QPs/RQs/SQs determines the amount of memory that is requested by the HCA for QP/RQ/SQ related contexts.

Memory allocation can be done in chunks of system pages. However, for performance reasons it is desirable to allocate a physically contiguous chunk of host memory for the Interconnect Context Memory (ICM).

6.1 Interconnect Context Memory

ICM is host memory that is allocated for the exclusive use of the HCA. The HCA uses the ICM to maintain and manage its control objects. The ICM is allocated upon HCA request at initialization (see [“Initialization” on page 243](#)). When the HCA needs system memory, it requests memory allocation from software and the latter should then allocate memory pages and pass them to HCA ownership. Software should not access these HCA owned pages until the HCA returns their ownership to SW.

The HCA asks software to allocate or release memory pages by posting events. SW should respond to these events using the `MANAGE_PAGES` command, see [Section 29.3.2, “MANAGE_PAGES - Driver Delivers Memory Pages for the Device Usage or Returns Pages,” on page 1662](#) for further details. Moreover, the HCA manages an independent ICM for each PCI function, enabling to shut down individual PCI functions without impacting others.

6.2 ICM Management in a Virtual Environment

This ICM shrink-and-grow capability can be used while moving virtual machines (along with their associated virtual HCAs).

The allocation of HCA resources to VMs can be performed in two ways:

- By virtualization software – manages allocation of HCA resources to VMs with very precise granularity (individual I/O channel), thereby using resources in an efficient and flexible manner.
- By host software – utilizes PCI SR-IOV semantics to allocate HCA resources to VMs.

PCI SR-IOV nomenclature distinguishes between Physical Functions (PFs) and Virtual Functions (VFs) with many-to-one relations between VFs and PFs. The key distinction between PFs and VFs is that PFs have a notion of HW ownership, meaning that a PF driver runs in a trusted environment, while VFs are logical partitions of the device and the VF driver cannot be trusted.

Bearing this distinction in mind, HCA associates the hardware related part of the ICM (FW area, physical interface parameters and the like) with PF(s), while the VM-related part of the ICM is associated with VFs via Guest Virtual Machine ID (GVMI). This allows implementing HCA virtualization at a hardware level. Each VM can manage its own virtual HCA, including allocating and de-allocating ICM memory for that virtual HCA. ICM memory allocated to virtual HCA

must be pinned by the Hypervisor (as well as WQs, CQs etc), so there will be an interaction between HCA and Hypervisor involving guest memory management.

Software can shutdown a guest VM by closing the HCA (DISABLE_HCA command). In response to this request, HCA will de-allocate all ICM memory associated with this VM and release all pages back to software.

The function reset of a VF results in erasing all entries associated with this VF from the BDF mapping table on the PCI interface. This prevents execution of any new PCI operation on behalf of this VF. It is expected that, before reset, all resources associated with this VF were already cleaned up and shut down. However, since the VF is not trusted, HW will also clean up all ICM-resident objects associated with this VF to ensure that a future enabling of this VF will not result in ghost control objects from the “previous life” of the VF.

The function reset of a PF results in wiping off all entries associated with this PF from the BDF mapping table on the PCI interface. This prevents execution of any new PCI operation on behalf of this PF. It is expected that, before reset, all VFs owned by that PF have already been shut down. Otherwise, shutting down the PF will cause a loss of I/O services to the VFs that are associated with it.

6.3 Memory Resources

To control its operation, the HCA uses various data structures, residing in host memory. To help the system developer correctly configure the system memory resources, this section presents an overview of the device’s data structures, and the memory resources required for HCA operation.

6.3.1 Address Translation Tables

Address translation tables are control structures in ICM allocated by the HCA for every memory key. These tables are used to translate virtual addresses to physical ones. They are programmed by software through the HCA command interface as well as by lightweight memory registration operations and are subsequently maintained by the HCA hardware.

For more details about address translation and protection please refer to [“Address Translation and Protection” on page 622](#).

6.4 HCA Control Objects (Contexts)

The HCA operation is controlled by “context objects” that are managed by the HCA in its ICM. For more details about control objects, please refer to [Chapter 8, “Software Interface” on page 248](#). The entry size of each entry depends on device configuration and is retrieved using the QUERY_ command. Software developers can use this table to get an idea of the control memory

footprint required for a given configuration of device capabilities. Each virtual HCA requires its own set of context memory.

Table 16 - ICM-Resident HCA Control Objects

Object	Entry size (bytes)	Max entries supported per virtual HCA	Note
Flow Interface (Queue Pair Context - QPC)	256	16M	Transport and basic Work Queues information
Extended RDMA-read Context (RdmaRdC)	64-2048	16M	Required for QPs responding to RDMA reads. One RdmaRdC is required for each RDMA read request that has been received but not yet processed. A fixed number of RdmaRdCs are allocated to each QP.
Shared Receive Queue Context	32	16M	SRQ context - location, size, control information
Completion Queue Context	64	16M	CQ contexts- location, size, control information
Events Queue Context	64	512	EQ contexts- location, size, control information
Memory Key Context	128	16M	Address space descriptors for data access
Memory Translation Table (MTT) - direct translation	8	unlimited	Virtual to Physical translation pages
MTT for indirect MKeys (or KLM)	16	unlimited	
Multicast Groups Identifier	64	unlimited	Multicast groups' information, QPs attached and so on
Persistent Signature Variable (PSV)	64	16M	Signature information for signature hand-over operations
Byte Stream Format (BSF)	64	unlimited	Signature operation control

6.4.1 Work Queues

Work (send and receive) Queues are implemented through two memory structures:

1. Queue Pair Context (QPC) - A control structure in ICM memory allocated by the HCA for every QP opened.
2. A pair of WQs, one for send and one for receive - A virtually-contiguous circular buffer accessible by user-level software to post send and receive WQEs that are read by hardware.

The Queue Pair Context (QPC) table contains Transport Context information and basic Work Queue (Send and Receive) information associated with each Queue Pair which is operating. The device supports up to 16M entries in the QPC table, and the actual number supported by a given

system configuration can be retrieved by the `QUERY_` command. For more details about WQs please refer to [Section 8.9.5, “UD Address Vector Format,” on page 362](#).

6.4.2 Shared Receive Queue Context

A Shared Receive Queue (SRQ) Context table contains information required by the HW in order to access the SRQ. The device supports 16M entries in the SRQ Context table. See [Section 8.10, “Shared Receive Queue \(SRQ\),” on page 365](#) for more details.

6.4.3 Completion Queues

CQs are implemented through two memory structures:

1. Completion Queue Context (CQC) - A control structure in ICM memory allocated by the HCA for every CQ opened.
2. Completion Queue - A virtually-contiguous circular buffer accessible by user-level software and used by hardware to post completions.

The Completion Queue Context (CQC) table contains information required by the HW in order to access the CQ, the CQ state information and so forth. The device supports 16M entries in the CQC table. For more details about CQ memory structures please refer to [Section 8.19, “Completion Queues,” on page 432](#).

6.4.4 Event Queues

EQs are implemented through two memory structure:

1. Event Queue Context (EQC) – A control structure in ICM memory allocated by the HCA for every EQ opened
2. Event Queue – A virtually-contiguous circular buffer accessible by system software and used by hardware to post event records - Event Queue Entries (EQEs).

The Event Queue Context (EQC) table contains information required by the HW in order to access the EQ, EQ state information and so on. The device supports 512 entries in the EQC table. For more details about EQs please refer to [Section 8.20, “Events and Interrupts,” on page 472](#).

6.5 User Access Region

The HCA provides a memory region that can be mapped directly to an un-trusted application by the OS and can be addressed by non-privileged code. This region is used to allow unprivileged applications to access needed HCA resources, such as ringing DoorBells, from userspace. Each process should get a distinct page mapped to this region to access HCA for data path operation, thereby isolating HCA access between the processes.

For more details about User Access Region format and usage please refer to [Section 8.2, “User Access Region \(UAR\),” on page 249](#).

7 Initialization and Teardown

7.1 Initialization

Adapter initialization is performed in the following three stages:

1. Device boot from attached NVMEM (Flash) – see [“Stage 1 - Device Boot From Attached NVMEM” on page 243](#).
2. PCI device initialization is performed via boot software enumeration. In this stage the device is discovered by the PCI enumeration software, and its memory BAR registers are programmed to enable the CPU access to the adapter’s configuration space. The device may include multiple functions. See [“Stage 2 - PCI Device Initialization via Boot Software Enumeration” on page 243](#).
3. Embedded switch or Physical port initialization.
4. Drivers initialization on each of the functions – see [“HCA Driver Start-up” on page 244](#).

7.1.1 Stage 1 - Device Boot From Attached NVMEM

After the HW reset signal is de-asserted, the device boots by reading the firmware image (configuration) from the attached Flash device. Each sector of the Flash is CRC-protected, and the device validates the signature of the sector read. If signature validation fails, the device initializes itself as a “Flash programming device”. Thus, once the host boots, the firmware image can be downloaded to the attached Flash device.

Once the firmware image has been loaded to the device’s internal memory and component initialization has completed, the device is ready to respond to PCI enumeration. Prior to this condition, the device responds with the ‘configuration retry status’ completion status to type0 configuration cycles targeting the device.

7.1.2 Stage 2 - PCI Device Initialization via Boot Software Enumeration

In this stage, the adapter device responds to configuration cycles allowing PCI enumeration software to discover the device. The device may contain multiple functions. Those functions may be Physical Functions (PFs) or Virtual Functions (VFs). Each function has a single BAR on the PCI address space.

The device can support the following PCI capabilities. The actual capabilities supported can be retrieved from the PCI configuration header. For more details, see the PCI Express Base Specifications.

- pcie
- msix
- sriov
- ari
- power management
- vpd

- serial number
- advanced error reporting

After PCI fabric initialization is completed (for example, BAR registers assigned), the HCA driver initializes and configures the device as described below.

Note: The driver should ensure that the Bus Master bit in the Command Register is set in the PCI configuration header of the HCA prior to executing further commands on the command interface.

7.2 HCA Driver Start-up

After PCI Express fabric enumeration is completed and software has assigned PCI memory space, the HCA is assigned with a single bar called “HCA BAR”, device bring-up can start. The device is initialized by executing the following sequence.

- Read the initialization segment from offset 0 of the HCA BAR, to retrieve the versions of the firmware and the command interface. The driver must match the command interface revision number. The format of the initialization segment is in [Table 14, “Initialization Segment,” on page 234](#)
- Write the physical location of the command queues to the initialization segment. If using 32-bit writes, write the most significant word first. The *nic_interface* field is part of the least significant word and must be set to zero (Full NIC/HCA driver), as are the *log_cmdq_size* and *log_cmdq_stride* fields.
- Read the *initializing* field from the initialization segment. Repeat until it is cleared (INIT_SEGMENT.initializing become 0).
- Execute ENABLE_HCA command.
- Execute QUERY_ISSI command. See [“ISSI - Interface Step Sequence ID” on page 248](#).
- Execute SET_ISSI command.
- Execute QUERY_PAGES to understand the HCA need for boot pages.
- Execute MANAGE_PAGES to provide the HCA with all required boot pages, The driver is allowed to give the sum of boot pages and *num_init_pages*.
- Execute QUERY_HCA_CAP to retrieve the device capabilities limits.
- Execute SET_HCA_CAP to modify system capabilities.
- Execute QUERY_PAGES to understand the HCA need for initial pages for executing commands. If *init_pages* is 0, no need to do the MANAGE_PAGES stage.
- Execute MANAGE_PAGES to provide the HCA with all require init-pages. This can be done by multiple MANAGE_PAGES commands.
- [\[EMC\] Set the Loopback Control Register.](#)
- [Execute SET_TUNNELED_OPERATION_PARAMS command if tunneled operations are supported](#)
- Execute INIT_HCA command to initialize the HCA.

- Execute SET_DRIVER_VERSION command (only in case HCA_CAP.driver_version==1). See [Section 29.3.19, “SET_DRIVER_VERSION,” on page 1829](#).
- Execute the “CREATE_EQ – Create EQ” on page 1880 command to create EQ. Map PAGE_REQUEST event to it.
- Execute QUERY_VPORT_STATE command to get vport state.
- For Ethernet, execute QUERY_VPORT_CONTEXT command to get permanent MAC address. (See [Section 14.1.5, “Virtual NIC Start-Up,” on page 843](#)).
- Execute MODIFY_VPORT_CONTEXT command to set current MAC address. (See [Section 14.1.5, “Virtual NIC Start-Up,” on page 843](#)).
- Map QP0 and QP1 to a receive WQ with the appropriate transport type. Prior to executing this command, software must create QPs to be used for Special QPs. Execute SET_MAD_DEMUX to choose which MADs will be forward to SW and which will be handled by the device.
- Set L2 addresses (mac_vlan), see [Section 29.19, “L2 Table Commands,” on page 2058](#)

The commands are explained in detail in [Chapter 29, “Command Reference” on page 1647](#).

7.3 HCA Driver Teardown and Re-initialization

Each Function of the HCA can be individually and independently shut down (and re-initialized/restarted later on) by software. When a PF is shut down, its VFs are also shut down. This operation is performed while the system shuts down gracefully, when PCI bus re-enumeration and memory re-allocation is required or when user decided to right-click-disable network device. In this case, software should perform the following steps:

- Stop all operations of function being shut down and destroy all resources.
- Unmap the EQ that the PAGE request events are mapped to and destroy the EQ.
- Execute the TEARDOWN_HCA command to close the HCA.
- Call MANAGE_PAGES in loop till all pages are reclaimed.
- Execute DISABLE_HCA command.

After TEARDOWN_HCA is acknowledged by the hardware, HCA will not attempt to access PCI on behalf of that ICM.

7.3.1 Fast Teardown

Fast teardown operation allows SW to perform teardown in faster way without the need to reclaim all the pages.

This feature is supported when HCA_CAP.fast_teardown==1.

To perform fast teardown, SW should:

- Perform TEARDOWN_HCA command with profile= prepare_fast_teardown.

- If operation success (state==0), then set INIT_SGEMENT.nic_interface=DISABLE
 - Poll INIT_SGEMENT.nic_interface until it get DISBALE value.
- If operation fail (state==1), Do the regular teardown flow.

7.4 Physical port Initialization and Configuration

The physical port managed, configured and controlled using port control registers, see [Table 951, “Ports Register Summary,” on page 1140](#).

Note: The second 4K block (with the clear_intx) register contain some more registers that can be read by SW without FW intervention (e.g., PTP time). Those are not exposed yet.

7.5 Wake On LAN (WOL)

This configurable mode allows to configure the relevant network port to wake-up an ACPI power-management based system. The wake operation is implemented by driving PCIe WAKE# signal low as long as PCIe PERST# signal is held low.

SW can configure one of the supported configuration options to wake-up the server using SET_WOL_ROL command (See [Section 29.3.34, “SET_WOL_ROL – Set WOL and ROL Mode,” on page 1846](#)).

The MAC address which is compared against it is the network port MAC address.

Tools can also configure WOL through the “ICMD_SET_WOL_ROL - Set Wake/Reset-On-LAN” command and read current configurations using “ICMD_QUERY_WOL_ROL - Get Wake/Reset-On-LAN”. The supported WOL modes are indicated by *wol_s/g/a/b/m/u/p* fields in “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities”.

7.6 Reset On LAN (ROL)

This configurable mode allows to configure the relevant network port to force a system reset, if the system supports ROL. The ROL operation is implemented by driving PCIe WAKE# signal low as long as PCIe PERST# signal is held high.

The user can configure one of the supported configuration options to reset the server using SET_WOL_ROL command (See [Section 29.3.34, “SET_WOL_ROL – Set WOL and ROL Mode,” on page 1846](#)).

The MAC address which is compared against it is the network port MAC address.

Tools can also configure ROL through the “ICMD_SET_WOL_ROL - Set Wake/Reset-On-LAN” command and read current configurations using “ICMD_QUERY_WOL_ROL - Get Wake/Reset-On-LAN”. The supported WOL modes are indicated by *rol_s/g* fields in “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities”.

7.7 ICM Management

For each of the functions, the device needs pinned memory allocated for its sole usage for proper operation. This memory, once assigned to the function is termed ICM.

The function can inform the driver that it needs more memory in two ways:

- Driver executes QUERY_PAGES command
- Function sends “Pages Request” event

In the response to QUERY_PAGES command or within the Pages Required event, includes information about the amount of additional memory that the device needs or the amount of spare memory that the function wants to return to the driver as it does not need it anymore.

If the function requires more memory, the driver should allocate memory and deliver it to the device using the MANAGE_PAGES command.

In response to “Pages Request” event, the driver must respond with at least a single MANAGE_PAGES command. Until receiving such a command, the device will not send additional event even if it needs more pages.

If the driver is not able to provide the HCA with memory, it will execute MANAGE_PAGES command appropriate opcode modifier. The device will terminate the commands with NO_RESOURCES syndrome.

If the function wants to release memory, the driver can reclaim memory using the MANAGE_PAGES command. It can repeat the MANAGE_PAGES commands until the number of returned pages is 0.

For the shared contexts, memory is requested from all PFs. The shared context is stored in the memory requested from one of the PFs and in case of reset to this PF, it will be copied to another PF.

7.7.1 Trust Model

Once pages are allocated to a function and delivered to the device, SW is not allowed to access them any more. Therefore, only trusted entity is requested to provide memory pages. The device will access only SW on physical functions of the device for pages because such SW is considered as trusted.

Whenever memory pages are required for a Virtual Function (VF), the device will request them from the driver of the PF associated with this VF.

In order to charge VFs for the ICM that they have been requested, the Hypervisor can install a “balloon driver” in each VF and reclaim pages from the VFs when the PF is requested to deliver pages on their behalf.

The option ROM drivers will need to use management commands while using the no-DRAM NIC interface. Such drivers will use icmd (TBD).

[Internal] The firmware should ignore such writes.

8 Software Interface

This chapter describes all the interfaces between the driver software and the device hardware, including specific control flows of the device. It contains the following sections:

- Section 8.1, “ISSI - Interface Step Sequence ID,” on page 248
- Section 8.2, “User Access Region (UAR),” on page 249
- Section 8.3, “Queue Pairs,” on page 254
- [Section 8.4, “RDB Context,” on page 278](#)
- Section 8.5, “DCT - Dynamically Connected Target,” on page 279
- Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284
- Section 8.7, “NIC_Vport Context - NIC Virtual Port Context,” on page 287
- Section 8.8, “HCA_Vport Context - HCA Virtual Port Context,” on page 295
- Section 8.9, “Work Queues,” on page 298
- Section 8.10, “Shared Receive Queue (SRQ),” on page 365
- Section 8.11, “XRC Shared Receive Queues,” on page 370
- Section 8.12, “Transport Interface Receive (TIR),” on page 373
- Section 8.13, “Transport Interface Send (TIS),” on page 381
- Section 8.14, “Receive Queue (RQ),” on page 383
- Section 8.16, “Send Queue (SQ),” on page 402
- Section 8.17, “Receive Memory Pool (RMP),” on page 411
- Section 8.18, “Flow Table,” on page 414
- Section 8.19, “Completion Queues,” on page 432
- Section 8.20, “Events and Interrupts,” on page 472
- Section 8.21, “XRQ Context,” on page 509
- [Section 8.22, “Transport Static Params Context,” on page 525](#)
- [Section 8.23, “TLS PROGRESS Params Context,” on page 528](#)
- [Section 8.25, “FPGA QP,” on page 530](#)
- [Section 8.26, “Transmit Scheduler,” on page 535](#)
- Section 8.28, “Command Interface,” on page 613
- Section 8.29, “ICMD Interface,” on page 619

8.1 ISSI - Interface Step Sequence ID

ISSI of this PRM revision is 1.

This parameter defines the step sequence ID of the interface between SW and the device. It is incremented by steps of 1.

ISSI is used to enable deprecating/modifying features, command interfaces as well as software, maintaining compatibility within the same ISSI value.

SW implementing this revision of the PRM should remember the ISSI of the PRM. In addition, during initialization flow (see Section 7.2, “HCA Driver Start-up,” on page 244), SW must perform a query of the supported ISSIs using QUERY_ISSI command (see Section 29.3.17, “QUERY_ISSI,” on page 1826), and then set the *actual_issi* informing the device on which ISSI to run, using SET_ISSI command (see Section 29.3.18, “SET_ISSI,” on page 1828).

It is clear that SW cannot run with an ISSI which is not supported by the device, so the *actual_issi* which software should set for the device must be = min (sw issi, max supported_issi).

Note that if QUERY_ISSI command returns with BAD_OPCODE, this indicates that the *supported_issi* is only 0, and there is no need to perform SET_ISSI.

8.1.1 ISSI History

This section describes the overall interface changes each time ISSI incremented.

8.1.1.1 ISSI = 1

- Added *cqe_version* to HCA_CAP (see Table 1571, “HCA Capabilities Layout,” on page 1671). SW must read this field to identify the CQE version.
- Deprecated ACTIVATE_TRACER and DEACTIVATE_TRACER commands.
- Deprecated SRQ commands. Instead, RMP and XRC_SRQ commands were added.
- Deprecated modifying srqn in RTS2RTS command (see Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919).

8.2 User Access Region (UAR)

The isolated, protected and independent direct access to the HCA HW by multiple processes is implemented via User Access Region (UAR) mechanism.

The UAR is part of PCI address space that is mapped for direct access to the HCA from the CPU. UAR is comprised of multiple pages, each page containing registers that control the HCA operation. UAR mechanism is used to post execution or control requests to the HCA. It is used by the HCA to enforce protection and isolation between different processes.

The cross-process isolation and protection is implemented through four key mechanisms:

1. Host SW maps different UAR pages to different consumers, hereby enforcing isolation between different consumers to access the same page in the HCA control space.
2. Each Control Object can be accessed (controlled) through a UAR page.
3. HCA driver associates a UAR page with a control object while initializing (“opening”) the object.
4. Prior to executing control operation on the object, HCA validates that the UAR page used to post the command matches the one specified in the contexts of that object.

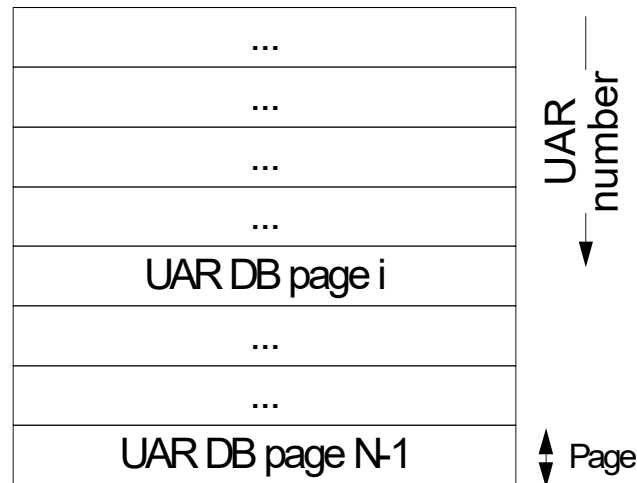
Operation is passed to the HCA by posting WQE to respective Work WQ, updating *Doorbell Record* (where applicable) and writing to respective *Doorbell Register* in the UAR page associated with that Work WQ. Writing to *Doorbell Register* of the HCA is further referred to as *ringing DoorBell*. *The DoorBell register in the UAR is the first 2 DWORDS of the blue-flame buffer.*

UAR pages can be mapped (on page granularity) to non-trusted applications. Hardware assures that bogus commands posted on the UAR bar will not harm anyone except the owner of the UAR BAR page this command was posted on.

8.2.1 UAR Sections

The UAR is implemented as a single BAR register per each function (Physical or Virtual) that contains multiple pages; page size should match the CPU (system) page size, and should be configured at device initialization. Figure 19 illustrates the structure of the UAR BAR.

Figure 19: HCA BAR Structure



8.2.2 UAR Page Format

Table 17 shows the layout of UAR page.

Table 17 - UAR Page Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										00h-1Ch
	cmds n						cmd	cq_ci																							
							cqn																								

Table 17 - UAR Page Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											28h-3Ch
																												40h				
eqn_with_arm								eqn_consumer_index_with_arm																				44h				
																												48h				
eqn								eqn_consumer_index																				04Ch-7FCh				
																												800-8FCh				
db_blueflame_buffer0_even																												900-9FCh				
db_blueflame_buffer0_odd																												A00-AFCh				
db_blueflame_buffer1_even																												B00-BFCh				
db_blueflame_buffer1_odd																												C00-CFCh				
db_blueflame_buffer2_even_fast_path																												D00-DFCh				
db_blueflame_buffer2_odd_fast_path																												E00-EFCh				
db_blueflame_buffer3_even_fast_path																												F00-FFCh				
db_blueflame_buffer3_odd_fast_path																																

Table 18 - UAR Page Format Field Descriptions

Offset	Bits	Name	Description	Access
20h	29:28	cmds_n	Command Sequence Number. This value should be '0 for the first DoorBell rung, and should be increment on each first DoorBell rung after a completion event. That is, $cmds_n = (num\_of\_completion\_event\_delivered + 1) \% 4$.	
	24	cmd	0x0: Notification_Solicited_UnSolicited - Request notification for next Solicited or Unsolicited completion event. cq_ci field should specify the CQ Consumer Counter at the time of the DoorBell ring. 0x1: Notification_Solicited - Request notification for next Solicited completion event. cq_ci field should specify the CQ Consumer Counter at the time of the DoorBell ring.	
	23:0	cq_ci	consumer index of the CQ.	
24h	23:0	cqn	CQ number	
40h	31:24	eqn_with_arm	EQ number to update its ci and arm	
	23:0	eqn_consumer_index_with_arm	consumer index with arm	
48h	31:24	eqn	EQ number to update its ci	
	23:0	eqn_consumer_index	consumer index	
800h-8FCh	2048	db_blueflame_buffer0_even	Doorbell blueflame register of buffer 0 even. See description below.	
900h-9FCh	2048	db_blueflame_buffer0_odd	Doorbell blueflame register of buffer 1 odd. See description below.	
A00h-AFCh	2048	db_blueflame_buffer1_even	Doorbell blueflame register of buffer 0 even. See description below.	
B00h-BFCh	2048	db_blueflame_buffer1_odd	Doorbell blueflame register of buffer 1 odd. See description below.	
C00h-CFCh	2048	db_blueflame_buffer2_even_fast_path	Doorbell blueflame register of buffer 0 even. See description below.	
D00h-DFCh	2048	db_blueflame_buffer2_odd_fast_path	Doorbell blueflame register of buffer 1 odd. See description below.	
E00h-EFCh	2048	db_blueflame_buffer3_even_fast_path	Doorbell blueflame register of buffer 0 even. See description below.	
F00h-FFCh	2048	db_blueflame_buffer3_odd_fast_path	Doorbell blueflame register of buffer 1 odd. See description below.	

Send DoorBells are rung in *blue flame* registers by writing the first 8 bytes of the WQE (that contains post counter, DS and QP/SQ number) to offset 0 of the blue flame register (See Table 42, "General - Ctrl Segment Format"). This post counter points to the beginning of the WQE to be executed. Note that this is different than the counter in the DB record which points to the next empty WQEBB. Send DBs of a QP/SQ can be rung on specific Blue Flame registers and cannot be interleaved on multiple Blue Flame registers.

Note that only QPs with the `latency_sensitive` property set are allowed to (and must) post BF on the fast-path BF buffers. QPs with the `latency_sensitive=0` should use the non fast path BF buffers.

Completion WQ DoorBells are rung in *CQ* register by writing CQ number, command, command sequence and CQ Consumer Index to respective fields. For best performance, this should be performed as a single 64-bit write operation.

Event WQ DoorBells are rung on the EQ register by writing the EQ number and the EQ consumer index. Writing to the first register will Arm the EQ. The second one will just update the consumer index without arming. When the EQ is armed, next event posted to the EQ will generate HW interrupt associated with this Event WQ. For details, refer to Table 244, "Event Queue Context Layout" on page 505

A recommended usage model, for getting interrupt whenever there is EQE in EQ is as follows: When arming the EQ at the first time, Consumer-index should be 0. When SW depletes the EQ from events, if it pop X events from the EQ, consumer-index should be incremental by X.

8.2.2.1 Blue Flame

BlueFlame is a low-latency mechanism that can be used to post latency-critical Send WQEs to the device. When BlueFlame is used, WQEs get written directly to a PCI BAR of the device (in addition to memory) so that the device may handle them without having to access memory, thus shortening the execution latency. For best performance, it is recommended to use BlueFlame when the HCA is lightly loaded. For high-bandwidth scenarios, it is recommended to use regular posting (without BlueFlame).

BlueFlame is supported for RC, DC, XRC, UC and UD transport services. **[Internal Ahmad TBD] Does it supported for all TSs ? Add ETH SQ or TIS.**

BlueFlame properties can be retrieved by querying the HCA_CAP. The BlueFlame is described through the following fields: *bf*, *log_bf_reg_size*, *log_max_bf_regs_per_page* and *log_max_bf_pages*.

The BlueFlame section contains equally-sized BlueFlame *registers*. Each such register contains a pair of equally-sized BlueFlame buffers. Each register (2 buffers) size is $2^{\log\_bf\_reg\_size}$ bytes. The layout of a BlueFlame page is shown in Table 17.

A BlueFlame buffer is used to post a Send Work Request. The WQE should be written directly to one of the BlueFlame register buffers as explained in "Posting a Work Request to Work Queue" on page 310. If WQE size is bigger than BlueFlame buffer size, then the WQE cannot be posted using BlueFlame. The two buffers in the BlueFlame register must be used alternately; in other words, *BlueFlame_Bufferi_even* should be used for each *even* posting to BlueFlame register *i*, and *BlueFlame_Bufferi_odd* should be used for each *odd* posting to the same register. Blue flame registers 2 and 3 are used for "fast path" posting.

When posted to a BlueFlame buffer, a WQE should be formatted as described in “Send WQE Format” on page 310.

BlueFlame register buffer must be written in chunks of DWORDs (aligned on 4 bytes) or multiple DWORDs. It may, though it is not recommended, be written out-of-order with respect to the offset of the posted writes. Different BlueFlame registers can be accessed by software simultaneously and independently. This way a single BlueFlame page can be used by multiple threads (permitted to access the same page) without locking. However, if software wishes to write different WQEs to the same BlueFlame register, it must use semaphores in order to avoid interleaving between the WQEs.

Blue Flame register should be written using Write Combining, generating bursts of one (or more) cache lines. If the WQE size is smaller than (a multiple of) a cache line, the WQE written to the Blue Flame buffer should be padded up to the closest cache line boundary. The HW uses the **DS** field of the WQE to determine the boundary of the WQE written to the Blue Flame buffer. Note that the WQE must be posted to the Work WQ *without* padding.

For regular (not BlueFlame) DoorBells, it is recommended not to use Write combining. The Doorbell and the Blue Flame registers are on the same UAR page. Therefore, it is recommended to map this UAR page twice. Once as write combining (for Blue Flame) and once as non Write Combining for regular DoorBells. For WQEs with DS=1, blue flame is not supported. DS=1 is the case for NOP or zero length send operations. The driver can either use regular posting (no blue flame) for this case or it can pad the WQE with extra zero length data segment as explained in “Send Data Segments” on page 339, thus extending DS to 2, which enables submission of blue flame on the WQE.

8.2.2.2 Sharing UARs

QP/SQ/RQ and CQ DoorBells require 64-bit writes. For best performance, it is recommended to execute the QP/SQ/RQ/CQ DoorBell as a single 64-bit write operation. For platforms that do not support 64 bit writes, it is possible to issue the 64 bits DoorBells through two consecutive writes, each write 32 bits, as described below:

- The order of writing each of the DWORDs is from lower to upper addresses.
- No other DoorBell can be rung (or even start ringing) in the midst of an on-going write of a DoorBell over a given UAR page.

The last rule implies that in a multi-threaded environment, the access to a UAR page (which can be accessible by all threads in the process) must be synchronized (for example, using a semaphore) unless an atomic write of 64 bits in a single bus operation is guaranteed. Such a synchronization is not required for when ringing DoorBells on different UAR pages.

8.3 Queue Pairs

8.3.1 Opening a Queue Pair

When opening a QP, software must create a QPC structure and pass it to the HCA with the CREATE_QP command (see “CREATE_QP - Create new QP” on page 1899). In return, HCA will return to the SW the QP number that was assigned to this QP that should be used for all further references to this QP.

Queue Pairs represent transmit and receive rings that can be used for the various transport services. The different transport services supported on Queue Pairs are:

- InfiniBand and RoCE: RC, UC, UD, XRC, DC
- InfiniBand UD QPs with stateless IPoIB/EoIB offloads
- and RoCE special QPs
- Raw Ethernet QPs

8.3.2 Modifying the Queue Pair Context

If the QPC needs to be modified (modify attributes, tear down, etc.) while it is owned by hardware, software needs to use the command queue to perform the required QP transition. The QP modification is guaranteed to be completed prior to completion of execution reported by the hardware. The command interface commands for handling QP modifications are listed in “[Command Reference](#)” on page 1647.

8.3.3 Path Migration

When executing path migration, the secondary address path configured in the QPC becomes a primary path, the original address path is dismissed and QP continues operation without a valid alternative path¹. Path migration can be initiated either by hardware (in which case the address path is changed transparently to the host software and event can be generated to notify the host in retrospect) or by software. In the latter case, the appropriate modify QP command needs to be executed on the command interface with the request to alter the path migration state.

8.3.4 Special QPs

8.3.4.1 Configuration and Ports De-multiplexing

The HCA can forward packets arriving on special QPs (QP0 and QP1) to the device driver. It is configured by “SET_MAD_DEMUX” on page 2100 to decide which of those packets will be forwarded to SW. For that purpose, the HCA creates two QPs per each network port of each virtual HCA. Device driver should retrieve from the HCA QP numbers allocated for this purpose and set them up using normal QP configuration steps. This is done with the “QUERY_MAD_DEMUX” on page 2096 commands.

8.3.4.2 Special QPs - Receive

Once special QPs are configured, all packets arriving on the special queue will be forwarded to the aliased RQ. Software can post receive WQEs to the alias RQ, which maintains the semantics of the UD RQ.

Packet handling in the HCA is the same as that of any other UD QP, i.e., only the payload is scattered to the addresses pointed to by the scatter list. The GRH, if present, is scattered to the first 40

1. Not having alternative path means that if main path fails, QP will be closed with error

bytes pointed to by the scatter list. If no GRH is present, the first 40 bytes are skipped. The overall packet length reported in the completion will be 40 bytes plus the length of the payload.

A completion will be reported to the CQ that the aliased QP is mapped to. If no WQEs are posted to the aliased RQ when the packet arrives, the packet will be silently dropped. Note that the completion opcode field for completions reported for packets arriving on QP1 is reserved. Hardware will validate and ensure that the incoming packets on QP1 have the Send opcode.

For the special alias QPs, hardware validates that QP0 packets arrive on VL15 and that QP1 packets arrive on a VL different from VL15 prior to sending (scattering) the packet payload to the aliased QP. If a VL violation is encountered, the packet is loudly dropped.

8.3.4.3 Mellanox Transport

An InfiniBand QP configured as MLX transport can be used to send raw data to the network. A QP that is configured as MLX transport does not generate a header in hardware, but gathers payload and sends it to the wire. No checks are performed, no acknowledge will be expected and no segmentation will be done (it is the software's responsibility to ensure the message does not exceed path MTU). The MLX transport QP has semantics similar to those of a UD QP, and CQ update will be scheduled after data transfer is scheduled on the link. MLX transport QPs can be used to send raw datagrams as well as for network debug purposes (for example, controlled injection of bogus packets). MLX transport QPs must be used only by privileged software.

8.3.4.4 Raw Ethernet Queue Pair

Raw Ethernet QPs can be created in order to send and receive Ethernet frames. These Queue Pairs can support multiple stateless offloads on RX and TX. Raw Ethernet Queue Pairs can be created with RX and TX Work Queue Buffers (rings), but they can also be created with a single work queue buffer (only RX or only TX).

On TX, the entire Ethernet frame is specified by the WQE. The headers of the frame are inlined into the WQE and the rest of the packet can be either inline or referenced by pointer. VLAN insertion is preformed by inlining the VLAN tag into the headers.

On RX, the entire Ethernet frame is scattered according to the receive WQE with few exceptions. VLAN offload can strip the VLAN tag and place it into the CQE, Header/Data Split can split the headers, and Inline Scatter can scatter packet headers into the CQE.

Further discussion of Ethernet offloads is discussed in [Chapter 4, “Networking and Stateless Offloads”](#) on page 185.

8.3.4.5 Multi User QP

This feature is supported only when `HCA_CAP.max_multi_user_group_size!=0`.

The multi-user QP capability enables managing a single Queue Pair (mostly single send work queue) from multiple processes/threads.

The main part of this capability is the ability to create and use a single Send queue, post WQEs to the Send queue from one (or more of the processes), but only execute the WQEs when ALL processes in the group agree to execute the work request(s) on the WQ.

The first phase is to create the Master QP with its work queues and transport properties (no limit on the transport type). Afterwards, different Slave QP of other processes join the group. By joining the group, each of the processes will have their own producer index to the work queue. Each process can update its producer index independently. The device calculates the global producer index to be the minimum of all producer indexes of the group members and execute work requests from the work queue based on this producer index.

(multi_user_qp_type=MULTI_USER_MASTER_QP and setting multi_user_group_size)

(multi_user_qp_type=MULTI_USER_SLAVE_QP and setting remote_qpn_or_multi_user_master_qp)

The flow includes the following steps:

- Create Master QP by setting multi_user_qp_type=MULTI_USER_MASTER_QP and setting multi_user_group_size.
- Transfer the Master QP number returned by the create_qp command above to the local processes which are members of the group. The way the QP number is communicated to the group members is out of the scope for this document, but one example could be using shared memory between the processes.
- Group members creates slave QPs using create_qp command with multi_user_qp_type=MULTI_USER_SLAVE_QP and setting the master QP number in remote_qpn_or_multi_user_master_qp.
 - Slave QP shall have no work queues, No_sq bit shall be set and Rq_type = 0x3 (zero size rq / no rq)
 - Slave QP shall have no transport attributes (AV, PSNs, CQ?)
- To use the group QP, one (or more) QPs shall write the WQE to the WQ and ring the doorbell to update the producer index (PI). The rest of the QPs shall only ring the doorbell to update the PI.

Note: All QPs within the group should share the same PD (TODO: and same UID?).

Note: CQEs are generated only to the master QP that is associated with a completion queue (CQ).

Tear Down Flow:

The tear down flow includes all the step from the setup flow but in the reverse order:

- All the processes (beside the master) should release all the resources (Memory, QP, etc.) [TODO: Shared PD problem - how to release shared resources?].

- The process that owns the master QP must release the QP after all resources of all the other processes have been released.

8.3.5 Queue Pair Context Format

The QPC structure is used to pass context information between SW and HW. The structure described in this section includes all context information. HW takes care of placing it in the right place in the ICM.

Table 19 shows context structure passed as an argument for QP configuration commands.

Table 19 - QP Context Structure Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset														
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0													0h										
state				lag_tx- port_ af- finity				st				ooo_enabled				sr_enable				pm_state				req_e2e_credit_mode				offload_ - type				end_padding_mode				multi_user_qp_type								
																																								pd				04h
mtu				log_msg_max								log_rq - size				log_rq - stride				no_sq				log_sq - size								rky				ulp_state- less_off- load_mode				08h				
counter_set_id																																				uar_page				0Ch				
				full handshake				cnak_re- verse_sl																												user_index				10h				
				log_page_size																																remote_qpn_or_multi_user_master_qp				14h				
																																								primary_address_path (See Table 21, “Address Path Structure Layout,” on page 274)				18h-40h

primary_address_path
(See Table 21, “Address Path Structure Layout,” on page 274)

Table 19 - QP Context Structure Layout

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset		
secondary_address_path (See Table 21, “Address Path Structure Layout,” on page 274)																													44h-6Ch					
log_ack_re q_freq						log_s- ra_max		extended_rnr_retry_valid				retry_ count		rnr_re- try		extended_retry_count_valid		fre		cur_rnr _retry		cur_re- try_ count		extend- ed_log_rn- r_retry					70h					
extended_cur_rnr_retry														packet_pacing_rate_limit_index															74h					
multi_user_group_size				next_send_psn																									78h					
		log_num_d- ci_stream_cha- nnels		cqn_snd																									7Ch					
		log_num_d- ci_er- rored_streams		deth_sqpn_or_initiator_src_dct																									80h					
				extended_retry_count																extended_cur_retry_count								84h						
				last_acked_psn																									88h					
				ssn																									8Ch					
				log_r- ra_max				atomic_ mode				rre		rwc		rae				page_offset								cd_slave_receive		cd_slave_send		cd_master		90h
		min_rnr_nak		next_rcv_psn																									94h					
				xrcd																									98h					
				cqn_rcv																									9Ch					

Table 19 - QP Context Structure Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset																			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										A0h-A4h																			
dbr_addr																												A0h-A4h																						
q_key																												A8h																						
reserved				rq_type				srqn_rmpn_xrqn																				ACH																						
								rmsn																				B0h																						
hw_sq_wqebb_counter														sw_sq_wqebb_counter														B4h																						
hw_rq_counter																												B8h																						
sw_rq_counter																												BC h																						
								roce_adp_retrans_rtt																				C0h																						
reserved														cqs		cs_req						cs_res								C4h																				
dc_access_key																												C8h-CCh																						
rdma active								comm_est		suspended		dbr_umem_valid																				send_msg_psn																		D0h
								rcv_msg_psn																				D4h																						
rdma_va																												D8h-DC h																						
rdma_key																												E0h																						
dbr_umem_id																												E4h																						

The fields of the QPC Structure are defined in Table 20.

Table 20 - QP Context Field Descriptions

Offset	Bits	Privilege	Name	Description	Access
0h	31:28	1	state	QP State (Valid only in QUERY_QP. In case QP.suspended==1, then valid also in CREATE_QP) 0x0: RST 0x1: INIT 0x2: RTR 0x3: RTS 0x4: SQEr 0x5: SQDRAINED - Send Queue Drained 0x6: ERR 0x7: SQDRAINING - Send Queue Draining	
	27:24		lag_tx- _port_af- finity	Define the desired TX LAG port number. Valid only when LAG is created by CREATE_LAG command or when HCA_CAP.lag_tx_port_affinity==1. If 0, the device will assign the affinity. See Chapter 14, “Virtualization” on page 838	
	23:16	1	st	Transport Service Type 0x0: RC 0x1: UC 0x2: UD 0x3: XRC 0x4: MLX 0x5: DCI 0x7: QP0 0x8: QP1 0x9: Raw_datagram 0xA: Raw_IPv6 0xC: REG_UMR 0x10: DC_CNAK Note that QPs of types QP0 and QP1 have the same properties as UD QP. There can be at most one for each type per port. QPs of types Raw Ethernet, Raw IPv6 have the same properties as MLX QP. There can be at most one for each type per port. QP of type REG_UMR have the same properties as UD QP. There can be at most one such QP per device. QP of types MLX , QP0, Raw Datagram are not supported with Roce.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
00h	15		<u>ooo_enabled</u>	<u>When set, the associated qp can receive out of order traffic</u> <u>Valid only for QUERY_QP command.</u>	
	14		<u>sr_enable</u>	<u>If set, selective repeat is enabled.</u> <u>Supported only when hca_cap.selective_repeat==1.</u> <u>Selective repeat - When a RoCE segment is transferred and lost,</u> <u>the device can either send all the segments from the lost one</u> <u>(go-back-n) or send the just the missing segment (selective</u> <u>repeat). Selective repeat might cause a reordering in the receive</u> <u>queue, hence the user shall wait for completion and not perform</u> <u>poll on memory. If a DCT is working in SR mode, it will be</u> <u>indicated in the ooo_enabled field. User can set DCT to work in</u> <u>SR or go-back-n mode in the SR_enable bit.</u>	
	12:11	N/ A	pm_state	Path migration FSM state 0x0: Armed 0x1: Rearm 0x3: Migrated Should be set to 0x3 for UD and DC QPs and for QPs which do not support APM.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
00h	9:8		req_e2e_credit_mode	Requester End to End Credit mode. indicates the e2e credit mode when is enabled (can be disabled by HW according to responder side). 0x0: msg_based 0x1: packet_based - Supported only when HCA_CAP.qp_packet_based==1. Need to make sure that responder side is configured with same mode.	
	7:4		offload_type	Offload type must be none or same as the XRQ.offload_type. 0x0: NONE - No offload. 0x1: RNDV- RNDV offload is enabled. Valid only when QP is connected to XRQ with RNDV offload. 0x2: NVMe-oF- NVMe-oF commands that are configured to be offloaded in the XRQ will be processed by the device. Valid only when QP is connected to XRQ with offload of NVMe-oF.	
	3:2	2	end_padding_mode	Scattering end of incoming send message (or raw Eth packet) 0x0: scatter_as_is 0x1: pad_to_cache_line_alignment Note that padding does not go beyond the receive WQE scatter entry length. [internal] 2 - pad to cache line alignment beyond scatter entry if needed. Note that end_padding_mode is reserved when QP is connected to RMP.	
	1:0		multi_user_qp_type	Multi user QP type. Supported only when HCA_CAP.max_multi_user_group_size!=0. 0x0: SINGLE_USER_QP - i.e. regular qp 0x1: MULTI_USER_MASTER_QP 0x2: MULTI_USER_SLAVE_QP	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
04h	31		wq_signature	If set, WQE signature will be checked on this QP. Not supported when QP is connected to XRQ.	
	30		block_lb_mc	Block loopback multicast. Support only if <i>block_lb_mc</i> =1 in HCA_CAP. See Table 1571, “HCA Capabilities Layout,” on page 1671.	
	29		atomic_like_write_en	If set, atomic_like_write_en is enabled on this QP	
	28		latency_sensitive	If set, this QP has latency_sensitive property	
	27		dual_write	Dual write QP	
	26		drain_siger	If set, a QP can be configured upon signature hand-over error to start the SQ draining process. See Section 17.3.14.2, “Signature Handover Completion Report,” on page 895.	
	25		multi_path	If set, the access to both requester’s and responder’s memory may be done in a different order than the order transmitted by the other party. Adaptive routing may be enabled for the request and response packets to enable fast convergence of congestion in the network. This field should be enabled in both QP parties in order to work efficiently. Supported only when HCA_CAP.multi_path_rc/dc/xrc_rdma==1 according to the transport service. Note: Data integrity offload is not supported in such QP.	
	23:0	1	pd	Protection Domain Reserved when st==REG_UMR.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
08h	31:29	1	mtu	0x1: BYTES_256 0x2: BYTES_512 0x3: BYTES_1K 0x4: BYTES_2K 0x5: BYTES_4K 0x7: RAW_ETHERNET_QP - must be used for a Raw Ethernet QP [TBD] Verify value 7 makes sense	
	28:24	0	log_msg_max	Maximum message size in bytes, allowed on the QP, is $2^{\log\_msg\_max}$. Must not exceed 30 for InfiniBand QPs.	
	22:19	1	log_rq_size	Receive WQ size is $2^{\log\_rq\_size}$ WQEs Reserved if <code>rq_type=srq, no_rq</code>	
	18:16	1	log_rq_stride	The size (stride) of a receive WQE equals $16 \cdot (2^{\log\_rq\_stride})$. The minimum size is 16 bytes. Max value for this field is 0x5. Reserved if: - <code>rq_type=srq, no_rq</code>	
	15		no_sq	No SQ (SQ size is 0). Can be set independent from <code>no_rq</code> . If <code>offload_type==RNDV</code> , then <code>no_sq</code> must be equal to 1.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
08h	14:11	1	log_sq_size	Send WQ size is $2^{\text{log\_sq\_size}}$ WQEBBs. Reserved if no_sq is set.	
	4	1	rlky	If set, this QP can use the Reserved L_Key for memory reference (physical address)	
	3:0		ulp_stateless_offload_mode	Sets ULP Stateless Offload mode for QP and can be supported only for UD QPs. See Section 4.6, “ULP Stateless Offloads,” on page 209. 0x0: None - No ULP Stateless Offloads are supported in this mode. 0x1: Basic_IPoIB - In this mode, QP supports basic IPoIB Stateless Offloads, limited to those offloads that are performed by IPoIB RQ/SQ. Supported only when QUERY_HCA_CAP.Ipoib_basic_offloads==1. The list of supported offloads in this mode can be queried through QUERY_HCA_CAP.op_mod = “IPoIB Basic Offload Capabilities”. Note - IPoIB RQ drops incoming packets with Send with Immediate header. IPoIB SQ does not support Send with Immediate and requires the extension Address vector. 0x2: Enhanced_IPoIB - In this mode, QP supports enhanced IPoIB Stateless Offloads. Supported only when HCA_CAP.ipoib_enhanced_offloads==1. See Section 4.6.1.2, “Enhanced ULP Offloads,” on page 210. Note that in this mode, these fields are reserved - end_padding_mode, wq_signature, block_lb_mc, latency_sensitive, drain_sigerr, log_rq_size, log_rq_stride, no_sq, log_sq_size, rlkey, counter_set_id, uar_page, user_index, fre, cqn_snd, page_offset, cd_slave_receive, cd_slave_send, cd_master, cqn_rcv, dbr_addr, rq_type, srqn/rmpn, hw_sq_wqebb_counter, sw_sq_wqebb_counter, hw_rq_counter, sw_rq_counter, cgs, cs_res, sl.	
0Ch	31:24		counter_set_id	Set of counters on which statistics on this QP are collected. counter_set should be allocated using ALLOC_Q_COUNTER command. See Section 29.35, “Q Counters Commands,” on page 2185. 0x0 means not connected to a counter set.	
	23:0	0	uar_page	UAR number allocated for ringing DoorBells for this QP (aliased to DoorBell and Blue Flame pages)	
10h	28		full_handshake	If set, DCI works in full handshake communication. i.e. wait for connection ack before sending the data. Supported only when HCA_CAP.dc_v2==1.	
	27:24		cnak_reverse_sl	Reverse sl for cnaks responses to the DCI. Supported only when HCA_CAP.dc_v2==1.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
10h	23:0		user_index	User_index - an opaque identifier which software sets, which will be reported to the Completion Queue. 0xffffffff - user_index does not exist (e.g like cq_version==0). Reserved when HCA_CAP.cq_version==0.	
14h	28:24		log_page_size	Log (base 2) of page size in units of 4KByte	
	23:0	1	remote_qpn_or_multi_user_master_qp	For connected transport services, indicates the Remote QP number. For DCI transport service, indicates the source DCT. Supported only when HCA_CAP.initiator_src_dct. This field can be reported in the CQE of the target side. When multi_user_qp_type==MULTI_USER_SLAVE_QP, indicates the Master QP number. otherwise reserved.	
18h-40h	352	1	primary_address_path (See Table 21, “Address Path Structure Layout,” on page 274)	These fields are used by connected services (RC/UC) to carry the primary path, Primary address path for the QP. Note that few of the parameters, for example pkey_index, are also used for UD QPs.	
44h-6Ch	352	1	secondary_address_path (See Table 21, “Address Path Structure Layout,” on page 274)	These fields are used by connected services (RC/UC) to carry alternate path. Alternative address path for the QP.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
70h	31:28	1	log_ack_req_freq	ACK request frequency. $2^{\text{log\_ack\_req\_freq}}$ is the frequency of issued ACK requests. Valid values are 0-8, giving an ACK request interval between 1-256 packets.	
	23:21	1	log_sra_max	$2^{\text{log\_sra\_max}}$ is the maximum number of outstanding Atomic/RDMA reads allowed as a requester	
	20	1	extended_rnr_retry_valid	If set, indicates working with extended_rnr_retry and extended_cur_rnr_retry instead of rnr_retry and cur_rnr_retry. Supported only when HCA_CAP.extended_rnr_retry==1.	
	18:16	1	retry_count	Transport timeout retry count. Valid only when extended_retry_count_valid==0	
	15:13	1	rnr_retry	RNR NAK retry limit prior to transition to error state. Valid only when extended_rnr_retry_valid==0.	
	12	1	extended_retry_count_valid	If set, indicates working with extended_retry_count and extended_cur_retry_count instead of retry_count and cur_retry_count. Supported only when HCA_CAP.extended_retry_count==1.	
	11	1	fre	FRWR enabled	
	10:8	1	cur_rnr_retry	Current RNR retry counter (QUERY only). The current RNR retry counter can vary from rnr_retry down to 1, where 1 means that the last retry attempt is currently executing. Valid only when extended_rnr_retry_valid==0.	
	7:5	1	cur_retry_count	Current transport retry counter (QUERY only). The current transport retry counter can vary from retry_count down to 1, where 1 means that the last retry attempt is currently executing. Valid only when extended_retry_count_valid==0	
	4:0	1	extended_log_rnr_retry	Log(base 2) of extended RNR NAK retry limit prior to transition to error state. Valid only when extended_rnr_retry_valid==1	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
74h	31:16	1	extend- ed_cur_rnr_retry	Extended Current RNR retry counter (QUERY only). The current RNR retry counter can vary from $2^{\text{extended\_log\_rnr\_retry}}$ down to 1, where 1 means that the last retry attempt is currently executing. Valid only when <code>extended_rnr_retry_valid==1</code>	
	15:0	1	pack- et_pac- ing_rate_li mit_index	Index into Rate Limit table. 0x0 means there is no rate limita- tion. Supported when <code>QoS_CAP==packet_pacing==1</code> . See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.	
78h	31:24		multi_us- er_group_- size	Indicates multi user group size (number of slaves that can be attached to this Master QP). Valid only when <code>multi_user_qp_type==MULTI_USER_MAS- TER_QP</code> , and must be >1 and in granularity of 2.	
	23:0	1	nex- t_send_psn	Next PSN to be sent	
7Ch	28:24		log_num_d ci_stream channels	<u>Log (base 2) of the number of dci stream channels.</u> <u>Must be <= HCA_CAP.log_max_dci_stream_channels.</u> <u>Valid only when st==DCI.</u>	
	23:0	1	cqn_snd	SQ completions are to be reported to this CQ number.	
80h	28:24		log_num_d ci_er- rored_strea ms	<u>Log (base 2) of the number of dci error stream channels before</u> <u>moving DCI to error.</u> <u>Must be <= HCA_CAP.log_max_dci_errored_streams.</u> <u>Valid only when st==DCI.</u>	
	23:0		deth_sqpn _or_initia- tor_src_dct	For UD QPs: Source QP number to use when sending data- grams in the DETH header. When set to zero, use the allocated QP number. Supported only when <code>HCA_CAP.set_de- th_sqpn==1</code> . For DCI transport service, indicates the source DCT. supported only when <code>HCA_CAP.initiator_src_dct</code> . This field can be reported in the CQE of the target side. For other transport service types, this field is reserved. See Table 1571, “HCA Capabilities Layout,” on page 1671.	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
84h	23:16	1	extended_retry_count	Extended transport timeout retry count. Valid only when extended_retry_count_valid==1	
	7:0	1	extended_cur_retry_count	Extended current transport retry counter (QUERY only). The current transport retry counter can vary from extended_retry_count down to 1, where 1 means that the last retry attempt is currently executing. Valid only when extended_retry_count_valid==1	
88h	23:0	1	last_acked_psn	The last acknowledged PSN for the requester (QUERY_QP only)	
8Ch	23:0	1	ssn	Requester Send Sequence Number (QUERY_QP only)	
90h	23:21	1	log_rra_max	2 ^{log_rra_max} is the maximum number of outstanding RDMA-read/Atomic operations allowed on the RQ	
	19:16	4	atomic_mode	If rae is set (RDMA atomic operations enabled), this field determines the level of atomic support. 0x1: IB_spec - IB spec 1.2.1 atomics only. All following enumerations support IB spec 1.2.1 atomics, as well as the specified masked atomic types (The supported sizes are reported in Atomic CAPs. 0x2: only_8B - support only 8B (CX compatible) 0x3: up_to_8B - (also 4B) 0x4: up_to_16B 0x5: up_to_32B 0x6: up_to_64B 0x7: up_to_128B 0x8: up_to_256B Value of 3 and up is also enabling LOAD Remote Micro Application atomics in responder with the respective size (size of Response).	
	15	1	rre	If set, RDMA read is enabled on the RQ	
	14	1	rwe	If set, RDMA write is enabled on the RQ	
	13	1	rae	If set, Atomic operations are enabled on the RQ	
	11:6	1	page_offset	Page offset in offset in quantas of (page_size / 64). For 4K pages, it is in 64B units.	
	2	1	cd_slave_receive	If set, Core Direct WQE processing mode is enabled on the receive queue of this QP	
	1	1	cd_slave_send	If set, Core Direct WQE processing mode is enabled on the send queue of this QP	
	0	1	cd_master	If set, Core Direct WQEs are supported on the send queue	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
94h	28:24	1	min_rnr_nak	Minimum RNR NAK timer value (TTTTT field encoding is according to the <i>InfiniBand Architecture Specification, Release 1.2, Vol. 1: 9.7.5.2.8</i>)	
	23:0	1	nex-t_rcv_psn	Next PSN (expected) to be received	
98h	23:0	1	xrcd	XRC Domain	
9Ch	23:0	1	cqn_rcv	RQ completions are to be reported to this CQ number. <i>Reserved when QP is connected to XRQ. CQEs will be generated on XRQ CQ.</i>	
A0h-A4h	64	1	dbr_addr	Physical address of DB Record See 8.9.2 Doorbell Record	
A8h	31:0	1	q_key	Q_Key to be validated against received datagrams. For QP1 transport type, q_key should be 0x80010000.	
ACh	26:24		rq_type	0x0: regular - (non zero size) rq 0x1: srq/rmp/xrc_srq/ <i>xrq</i> 0x3: zero_size_rq - (no_rq) This field must be 1 for DC and XRC service types.	
	23:0	1	srqn_rmpn_xrqn	SRQ/RMP/XRQ number this QP is associated with. Reserved for XRC QP.	
B0h	23:0	1	rmsn	Responder's current message sequence number (QUERY_QP only)	
B4h	31:16		hw_sq_wq_ebb_counter	Current HW SQ WQEBB counter. Points to the next WQE to be consumed by HW (QUERY_QP only)	
	15:0		sw_sq_wq_ebb_counter	Current SW SQ wqebb counter. Points to the next WQE to be produced by SW, as kept in the HW QPC (QUERY_QP only).	
B8h	31:0		hw_rq_counter	Current HW RQ WQE index Points to the next WQE to be consumed by HW). Bits [31:16] are reserved. (QUERY_QP only). Reserved if rq_type== <i>srq</i> or no_rq	
BCh	31:0		sw_rq_counter	Current SW RQ WQE index points to the next WQE to be produced by SW) as kept in the HW QPC. Bits [31:16] are reserved. (QUERY_QP only). Reserved if rq_type==srq or no_rq	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
C0h	23:0		roce_ad- p_re- trans_rtt	The current round trip time in microsec. Supported when HCA_CAP.roce_accl=1. Valid in QUERY_QP command only.	
C4h	16		cgs	UD responder only 0 - GRH (if present) is scattered to beginning of receive buffer. Regardless of GRH presence, data will be scattered starting from an offset of 40 bytes. 1- GRH (if present) is scattered to CQE. Data will be scattered to start of receive buffer (or to CQE depending on data size and cs_res field value as described below). Supported only when HCA_CAP.sctr_data_cqe==1. Supported only when HCA_CAP.sctr_data_cqe==1.	
	15:8		cs_req	CQE scatter requester - scatter read and atomic responses to the CQE 0x0: disable - data is always scattered according to send WQE.scatter list 0x11: up_to_32B - if payload<=32B, data is scattered into the CQE (inline). Supported only when HCA_CAP.sctr_data_cqe==1. For DC support only when HCA_CAP.dc_req_sctr_data_cqe==1. 0x22: up_to_64B - if payload<=64B, data is scattered into the CQE (inline). Supported only when HCA_CAP.sctr_data_cqe==1.	
	7:0		cs_res	CQE scatter responder - Scatter the responder packet data to the CQE 0x0: disable - data will always be scattered to the receive buffer 0x01: up_to_32B - if payload<=32B, data is scattered into the CQE (inline). N/A for UD and DC. Supported only when HCA_CAP.sctr_data_cqe==1. 0x02: up_to_64B - if payload<=64B, data is scattered into the CQE (inline). For UD, this mode can only be selected when cgs=1	
C8h- CCh	64		dc_ac- cess_key	DC access key	

Table 20 - QP Context Field Descriptions (Continued)

Offset	Bits	Privilege	Name	Description	Access
D0h	31		rdma_active	Indicates whether the responder is currently processing an incoming RDMA-W. Valid only when suspended==1.	
	30		comm_est	Indicates whether the “communication established” event was raised for this QP. When creating a suspended QP, this bit is valid only when the QP state is RTR. Valid only when suspended==1.	
	29		suspended	Indicates whether the QP is suspended. Valid only for QUERY_QP and CREATE_QP. Supported only when HCA_CAP.suspended_qp==1.	
	28		dbr_umem_valid	Indicates if dbr address is given directly or by umem_id and umem_offset. When 0, dbr_addr allocates the physical address of the dbr. When 1, dbr_addr allocates the umem_offset of the dbr, and umem_offset is valid	
	23:0		send_msg_psn	The first PSN of the current executing WQE in the Send Queue. Valid only when suspended==1.	
D4h	23:0		rcv_msg_psn	The first PSN of the current executing WQE in the Receive Queue or the current executing RDMA-W operation at the responder. Valid only when suspended==1.	
D8h-DCh	64		rdma_va	The initial virtual address of the RDMA-W operation associated with the next expected RDMA-W packet at the responder. Valid if rdma_active=1. Valid only when suspended==1.	
E0h	31:0		rdma_key	The RKey associated with the next expected RDMA-W packet at the responder. Valid if rdma_active=1. Valid only when suspended==1.	
E4	31:0		dbr_umem_id	UMEM ID of DB Record. Valid only when dbr_umem_valid==1. when set, dbr_addr indicates umem_offset instead of physical address of the DB Record.	

Table 21 shows the address path structure format .

Table 21 - Address Path Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
free_ar																pkey_index											00h					
								grh	mlid							rlid											04h					
ack_timeout									src_addr_index							log_rtm			stat_rate			hop_limit					08h					
				tclass							flow_label															0Ch						
rgid_or_rip																															10h-1Ch	
				f dscp		f ecn		f eth_prio		ecn		dscp					udp_sport											20h				
eth_prio		sl				vhca_port_num							rmac[47:32]											24h								
rmac[31:0]																															28h	

Table 22 details address path fields.

Table 22 - Address Path Structure Fields

Offset	Bits	port (IB)	Name	Privilege	Description	Access
000h	31	IB/ ETH	fl	1	Force loopback. ~~For RoCE and when HCA_CAP.fl_rc_qp_when_roce_disabled==1, RC QP can be created with fl==1 even if RoCE is disabled.~~ For RoCE and when HCA_CAP.fl_rc_qp_when_roce_disabled==1, RC QP can be created, and move to RTR while setting fl==1 even if RoCE is disabled. On such case source_addr_index, rgid_or_rip and rmac should be reserved.	
	30	IB/ ETH	free_ar		When set, packets sent using this QP are marked for free adaptive routing. On IB wire, it is marked in LRH and BTH. For RoCE, it is marked in BTH. Supported only for UD and UC transports.	
	15:0	IB	pkey_index	0	Index to pkey table Reserved for RoCE. For RoCE index 0, which contains the default PKEY: 0xFFFF, is chosen automatically	

Table 22 - Address Path Structure Fields (Continued)

Offset	Bits	port (IB)	Name	Privilege	Description	Access
004h	23	IB	grh	0	Use GRH For InfiniBand, if set, GRH will be placed in the packet header, if clear, no GRH. Reserved for RoCE.	
	22:16	IB	mlid	0	LMC bits of LID Reserved for RoCE.	
	15:0	IB	rlid	1	Remote LID. Reserved when fl==1. Reserved for RoCE.	
008h	31:27	IB/ ETH	ack_timeout	1	Ack timeout timer for activation of retransmission mechanism. If zero - timeout is disabled. timeout is $4.096 \mu S * 2^{(ack\_timeout)}$	
	23:16	IB/ ETH	src_addr_index	0	For InfiniBand: Index to Address Table in the port to get source GID address. For RoCE: index to Address table to get the source GID/IP and MAC address, VLAN ID, IP Type (IPv4, IPv6) and RoCE type.	
	15:12	IB/ ETH	log_rtm	1	Retry Timeout Multiplier. ACK timeout will be multiplied by 2^{log_rtm} every consecutive retry. See “Exponential Backoff Timer”	
	11:8	IB/ ETH	stat_rate	0	Maximum static rate control. 0: no_limit - no limit on static rate (100% port speed) 1-6: reserved 7: Gbps_2.5 8: Gbps_10 9: Gbps_30 10: Gbps_5 11: Gbps_20 12: Gbps_40 13: Gbps_60 14: Gbps_80 15: Gbps_120 17: Gbps_56 20: Gbps_25 21: Gbps_100 22: Gbps_200 25: Gbps_50	
	7:0	IB/ ETH	hop_limit	1	For IB and RoCE version 1.0: GRH Hop Limit. For RRoCE: IPv6 hop limit or IPv4 TTL	

Table 22 - Address Path Structure Fields (Continued)

Offset	Bits	port (IB)	Name	Privilege	Description	Access
00Ch	27:20	IB/ETH	tclass	1	For InfiniBand and RoCE v1.0: GRH TClass	
	19:0	IB/ETH	flow_label	1	IPv6 Flow Label	
10h-1Ch	128	IB/ETH	rgid_or_rip		Remote GID/IP address. For IB port - Valid if the GRH bit is set. For an Ethernet port for RoCE mode, this field must always be valid and contain the remote GID address. For an Ethernet port for R-RoCE mode, this field contains IPv4 or IPv6 depending on the IP type which appears in RoCE Address table entry pointed by src_addr_index field: for IP type == IPV4 then rgid/rip bits [31:0] specify the remote IPv4 address and rgid/rip bits [127:32] are reserved, when L3 header type == IPV6 rgid/rip bits [127:0] contains an IPv6 address.	

Table 22 - Address Path Structure Fields (Continued)

Offset	Bits	port (IB)	Name	Privilege	Description	Access
20h	27	ETH	f_dscp		Force IP DSCP for UD/DC QPs. Take IPv4/IPv6 DSCP field from QPC and not from send WQE. When QP can generate only RoCE v.1 packets this field is reserved.	
	26	ETH	f_ecn		Force IP ECN only for UD/DC RoCE QPs. Take IP ECN field from QPC and not from send WQE. When QP can generate only RoCE v.1 packets this field is reserved.	
	24	ETH	f_eth_prio		Force Ethernet user priority for UD/DC RoCE QPs. When set take Ethernet priority from QP's Address Path and not from send WQE. Note: When f_eth_prio is enabled but send WQE points to RoCE Address Table entry with vlan_valid disabled, priority tagged frame is created and Ethernet priority and DEI/CFI fields are taken from QP's Address Path.	
	23:22	ETH	ecn		For R-RoCE: IP ECN This field is reserved when QP supports only RoCE v1.0.	
	21:16	ETH	dscp		For R-RoCE: IP DSCP This field is reserved when QP supports only RoCE v1.0.	
	15:0	ETH	udp_sport		For R-RoCE v2.0 RC/UC QPs: UDP source port. must belong to the range: [QUERY_HCA_CAP.r_roce_udp_src_port_range_min-QUERY_HCA_CAP.r_roce_udp_src_port_range_max]. Valid only when HCA_CAP.Roce_CAP.sw_r_roce_src_udp_port==1. See Section 1571, "HCA Capabilities Layout," on page 1671	
024h	30:28	ETH	eth_prio		Ethernet Priority (PCP). Reserved for InfiniBand.	
	27:24		sl		InfiniBand: Service Level (SL)	
	23:16		vhca_port_num		VHCA port number	
	15:0	ETH	rmac[47:32]		Upper bits of remote MAC address. Valid only for RoCE QPs.	

Table 22 - Address Path Structure Fields (Continued)

Offset	Bits	port (IB)	Name	Privilege	Description	Access
28h	31:0	ETH	rmac[31:0]		Lower bits of remote MAC address. Valid only for RoCE QPs.	

8.4 RDB Context

RDB reflects the latest processed RDMA-R or atomic operation. The responder can never be in the middle of atomic processing – it is assumed that while the QP was suspended, a response was sent or assumed to be silently dropped. However, the responder can be in the middle of RDMA-R processing. In this case, the responder does not initiate responses upon resuming the QP. Rather, the QP peer is assumed to retry the operation and the RDB classifies it as a duplicate RDMA-R request and processes it accordingly.

RDB context structure is shown in Table 23.

Table 23 - RDB Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												type	00h			
																													04h			
								psn																				08h				
rkey																												0Ch				
address																												10h-14h				
byte_count																												18h				
																												1Ch				
atomic_resp																												20h-3Ch				

Table 24 - RDB Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	3:0	type	RDB type 0: Invalid - used to mark entries were never used. 1: RDMA response. 2: Atomic response.	

Table 24 - RDB Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
08h	23:0	psn	PSN on the original request.	
0Ch	31:0	rkey	RKEY of the request.	
10h-14h	64	address	Address specified in the request.	
18h	31:0	byte_count	The request transaction size. For atomics, this field should be 8 for standard atomics.	
20h-3C	256	atomic_resp	The MSBs of the atomic response data starting from offset 0.	

8.5 DCT - Dynamically Connected Target

8.5.1 DCT Context

DCT context structure is shown in Table 25.

Table 25 - DCT Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
ooo enabled		sr enable				state																	offload_type								00h					
								user_index																				04h								
								cqn																				08h								
counter_set_id								atomic_-mode		rre		rwe		rae		atomic like write en		latency sensitive		rlky		force full handshake		multi path												0Ch
								cs_res										min_rnr_nak												10h						
								srqn_or_xrqn																				14h								
								pd																				18h								
tclass												flow_label																1Ch								
dc_access_key																												20h-24h								

Table 25 - DCT Context Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
					mtu			port							pkey_index															28h		
								my_addr_index													hop_limit					2Ch						
dc_access_key_violation_count																																30h
																					eth_prio		ecn		dscp				34h			
								min_guaranteed_connection																								38h
								max_outstanding_connections																								3Ch
ece																																40h
																																44h-6Ch

Table 26 - DCT Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	ooo_en-abled	<u>When set, the associated qp can receive out of order traffic</u> <u>Valid only for QUERY_QP command.</u>	
	30	sr_enable	If set, selective repeat is enabled. <u>Supported only when hca_cap.selective_repeat==1.</u> <u>Selective repeat - When a RoCE segment is transferred and lost,</u> <u>the device can either send all the segments from the lost one (go-</u> <u>back-n) or send the just the missing segment (selective repeat).</u> <u>Selective repeat might cause a reordering in the receive queue,</u> <u>hence the user shall wait for completion and not perform poll on</u> <u>memory. If a DCT is working in SR mode, it will be indicated in</u> <u>the ooo_enabled field. User can set DCT to work in SR or go-</u> <u>back-n mode in the SR_enable bit.</u>	
	27:24	state	DCT State 0x0: Active 0x1: Draining 0x2: Drained valid only for QUERY_DCT command	
	7:4	offload_-type	0x0: NONE - No offload. 0x1: RNDV- RNDV offload is enabled. Valid only when DCT is connected to XRQ with RNDV offload. 0x2: NVMe-oF- NVMe-oF commands that are configured to be offloaded in the XRQ will be processed by the device. Valid only when QP connected to XRQ with offload of NVMe-oF.	
04h	23:0	user_index	User_index - an opaque identifier which software sets and which will be reported to completion queue. 0xffffffff - user_index does not exist (e.g like cq_version==0). Reserved if HCA_CAP.cq_version==0.	
08h	23:0	cqn	Completions will be reported to this CQ number. Reserved when DCT connected to XRQ, CQEs will be generated on XRQ CQ and not the QP CQ.	

Table 26 - DCT Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
0Ch	31:24	counter_set_id	Set of counters on which statistics on this DCT are collected. counter_set should be allocated using ALLOC_Q_COUNTER command. See Section 29.35, “Q Counters Commands,” on page 2185. 0x0 means not connected to a counter set.	
	23:20	atomic_mode	If rae is set (RDMA atomic operations enabled), this field determines the level of atomic support. 0x1: IB_SPEC- IB spec 1.2.1 atomics only. All following enumerations support IB spec 1.2.1 atomics, as well as the specified masked atomic types. 0x2: ONLY_8B - support only 8B 0x3: UP_8B - up to 8B (also 4B) 0x4: UP_16B - up to 16B 0x5: UP_32B - up to 32B 0x6: UP_64B - up to 64B 0x7: UP_128B - up to 128B 0x8: UP_256B - up to 256B	
	19	rre	If set, RDMA read is enabled on this DCT	
	18	rwe	If set, RDMA write is enabled on this DCT	
	17	rae	If set, Atomic operations are enabled on this DCT	
	16	atomic_like_write_en	If set, atomic_like_write_en is enabled on this DCT	
	15	latency_sensitive	If set, this DCT has latency_sensitive property	
	14	rlky	If set this DCT can use the Reserved L_Key for memory reference (physical address)	
	13	force_full_handshake	Force full handshake.supported only when HCA_CAP.dc_v2==1. 0x0: no_force - half handshake speculative data will be accepted by the dct. 0x1: force - half handshake communication requests to this dct will be downgraded to full handshake and speculative data won't get accepted.	
	12	multi_path	If set, the access to both requester's and responder's memory may be done in a different order than the order transmitted by the other party. Adaptive routing may be enabled for the request and response packets to enable fast convergence of congestion in the network. This field should be enabled in both DCI and DCT parties in order to work efficiently. Supported only when HCA_CAP.multi_path_dc_rdma ==1.	

Table 26 - DCT Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
10h	23:16	cs_res	CQE scatter responder - Scatter the responder packet data to the CQE 0x0: disable - data will always be scattered to the receive buffer 0x2: up_to_64B - if payload<=64B, data is scattered into the CQE (inline).	
	12:8	min_rnr_nak	Minimum RNR NAK timer value (TTTTT field encoding is according to the <i>InfiniBand Architecture Specification, Release 1.2, Vol. 1: 9.7.5.2.8</i>)	
14h	23:0	srqn_or_x-rqn	SRQ/XRQ number this DCT is associated with.	
18h	23:0	pd	Protection Domain	
1Ch	31:24	tclass	For InfiniBand and RoCE v1.0: GRH TClass This field is reserved for RoCE DCT which does not support RoCE v1.0.	
	19:0	flow_label	IPv6 Flow Label	
20h-24h	64	dc_access_key	DC access key	
28h	26:24	mtu	0x1: BYTES_256 0x2: BYTES_512 0x3: BYTES_1K 0x4: BYTES_2K 0x5: BYTES_4K	
	23:16	port	port number	
	15:0	pkey_index	Index to pkey table Reserved for RoCE. For RoCE index 0, which contains the default PKEY: 0xFFFF, is chosen automatically.	
2Ch	23:16	my_addr_index	For InfiniBand, Index to GID in the port to get source GID address. For RoCE, index of entry in port's RoCE Address table, which sets the source GID/IP, MAC addresses, VID (if available), L3 version and RoCE protocol version of this DCT.	
	7:0	hop_limit	For InfiniBand and RoCE: GRH Hop Limit For RRoCE: IPv6 hop limit or IPv4 TTL	
30h	31:0	dc_access_key_violation_count	DC access key violation counter. valid for QUERY_DCT command only.	

Table 26 - DCT Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
34h	10:8	eth_prio	Ethernet Priority (PCP). Reserved for IB.	
	7:6	ecn	For R-RoCE: IP ECN This field is reserved when DCT supports RoCE v1.0. Reserved for IB	
	5:0	dscp	For R-RoCE: IP DSCP This field is reserved when DCT supports RoCE v1.0. Reserved for IB.	
38h	23:0	min_guaranteed_connection	Number of connections guaranteed to be serviced in parallel. Supported only when HCA_CAP.dc_v2==1.	
3Ch	23:0	max_outstanding_connections	Maximum number of connections could be served in parallel. Supported only when HCA_CAP.dc_v2==1.	
40h	31:0	ece	<u>Enhanced Connection Establishment provided by device.</u> <u>0 means invalid.</u> <u>Supported when HCA_CAP.ece = 1</u>	

8.6 Esw_Vport Context - E-switch Virtual Port Context

For more details on esw_vport context see Section 14.1.2.2, “Vport Context,” on page 841.

Table 27 presents vport context structure.

Table 27 - Esw_vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
fdb to vport reg c				vport svlan strip	vport cvlan insert	fdb_to_vport_reg_c_id																												00h
																																04h		
svlan cfi	svlan_pc p		svlan_id												cvlan cfi	cvlan_pc p		cvlan_id														08h		
																																0Ch-ECch		

Table 27 - Esw_vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sw_steering_vport_icm_address_rx																																F0h-F4h
sw_steering_vport_icm_address_tx																																F8h-FCh

Table 28 - Esw_vport Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	fdb_to_vport_reg_c	If set, this Esw vport supports getting Register C metadata from FDB.	
	28	vport_svlan_strip	When set, strip cvlan of received packet if exist. supported only when ESWITCH_CAP.vport_svlan_strip==1. See Table 1593, “E-switch Capabilities Layout,” on page 1768	
	27	vport_cvlan_strip	When set, strip svlan of received packet if exist. supported only when ESWITCH_CAP.vport_cvlan_strip==1. See Table 1593, “E-switch Capabilities Layout,” on page 1768	
	26	vport_svlan_insert	When set, insert svlan for transmitted packets. supported only when ESWITCH_CAP.vport_svlan_insert==1. See Table 1593, “E-switch Capabilities Layout,” on page 1768	
	25:24	vport_cvlan_insert	cvlan insertion for transmitted packets 0x0: no_insert 0x1: insert_when_no_cvlan - insert cvlan only if there is no cvlan in packet. Supported only when ESWITCH_CAP.vport_cvlan_insert_if_not_exist==1. 0x2: insert_or_overwrite - insert cvlan if there is no cvlan, else overwrite the VLAN ID (cvlan_id). supported only when ESWITCH_CAP.vport_cvlan_insert_overwrite==1. 0x3: insert_always - insert cvlan regardless of the existence or absence of vlan and its tpid. Mutual exclusive with vport_svlan_insert. Supported only when ESWITCH_CAP.vport_cvlan_insert_always==1	
	23:16	fdb_to_vport_reg_c_id	Bit map per register_c ID indicating whether this register is passed as metadata to Vport NIC Rx Flow Table. Only Registers enabled by QUERY_HCA_CAP.fdb_to_vport_reg_c_id may be used. Valid only if fdb_to_vport_reg_c is set. Set bit means this register is passed. Note that the bit mask must define a range of C Registers, starting at C Register 0. i.e. bits must be set continuously starting from 0. Bit 0: REG_C_0 Bit 1: REG_C_1 Bit 2: REG_C_2 Bit 3: REG_C_3 Bit 4: REG_C_4 Bit 5: REG_C_5 Bit 6: REG_C_6 Bit 7: REG_C_7	

Table 28 - Esw_vport Context Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	svlan_cfi	svlan_cfi to be inserted to transmitted packet. Valid only when vport_svlan_insert==1.	
	30:28	svlan_pcp	svlan_pcp to be inserted to transmitted packet. valid only when vport_svlan_insert==1.	
	27:16	svlan_id	svlan_id to be inserted to transmitted packet. valid only when vport_svlan_insert==1.	
	15	cvlan_cfi	cvlan_cfi to be inserted to transmitted packet. valid only when vport_cvlan_insert!=0.	
	14:12	cvlan_pcp	cvlan_pcp to be inserted to transmitted packet. valid only when vport_cvlan_insert!=0.	
	11:0	cvlan_id	cvlan_id to be inserted to transmitted packet. valid only when vport_cvlan_insert!=0.	
F0h-F4h	64	sw_steering_vport_icm_address_rx	ICM address in FDB Rx steering domain Sending the packet to this VPort. valid only if FDB table supports SW owner, and the value is non-zero.	
F8h-FCh	64	sw_steering_vport_icm_address_tx	ICM address in FDB Tx steering domain Sending the packet to this VPort. valid only if FDB table supports SW owner, and the value is non-zero.	

8.7 NIC_Vport Context - NIC Virtual Port Context

For more details on NIC_Vport context see Section 14.1.2.2, “Vport Context,” on page 841.

Note that this context is supported only when HCA_CAP.eth_virt==1. [Table 1571, “HCA Capabilities Layout,”](#) on page 1671.

Table 29 presents Vport context structure.

Table 29 - NIC_Vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
multi_prio_sq						min_wqe - inline_ mode																					00h	roce_en				
																											disable_uc_local_lb		disable_mc_local_lb			

Table 29 - NIC_Vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																				
arm change event																										event on mtu		event on promise change	event on vlan change	event on mc address change	event on uc address change	04																				
													affiliation - criteria		affiliated_vhca_id											08h																										
																																		0Ch-20h																		
																	mtu																	24h																		
system_image_guid																																28h-2Ch																				
port_guid																																30h-34h																				
node_guid																																38h-3Ch																				
max_qp_retry_limit																max_qp_retry_period																40h																				
																																44h-5Ch																				
qkey_violation_counter																																68h																				
																																6Ch-ECh																				
promise uc		promise mc		promise all				allowed_list_type												allowed_list_size										F0h																						
																														F4h-F8h																						
																				(See Table 33, “MAC Address Layout,” on page 293)																																F4h-F8h FCh
																				sw_network_metadata																																

Table 29 - NIC_Vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
current_uc_mac_address[0]/current_mc_mac_address[0]/vlan[0]																																100h-104h
current_uc_mac_address[1]/current_mc_mac_address[1]/vlan[1]																																108h-10Ch
...																																120h-...

Table 30 - NIC_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	multi_prio_sq	If set, the one or more of the SQs will be transmitting packets of different priorities. By default this field is 0. Supported only when HCA_CAP.multi_prio_sq==1	
	26:24	min_wqe_inline_mode	Sets the minimal allowed inline mode for NIC Vport SQs. Each SQ has inline mode that defines the minimal required inline headers in Eth Segment of the SQ's WQEs. Note: SQ.min_wqe_inline_mode must be >= Nic_vport.min_sq_wqe_inline_mode. Must be configured before the VF is loaded. For more details - See Section 8.9.4.1.5, "Eth Segment and Padding Segment," on page 318. Valid only when HCA_CAP.wqe_inline_mode == 1	
	2	disable_mc_local_lb	If set, local loopback for multicast traffic is disabled. Supported only when HCA_CAP.disable_local_lb==1. Invalid for ROCE.	
	1	disable_uc_local_lb	If set, local loopback for unicast traffic is disabled. Supported only when HCA_CAP.disable_local_lb==1. Invalid for ROCE.	
	0	roce_en	0 - roce is disabled on this vport 1 - roce is enabled on this vport Till RoCE is enabled (nic_context.roce_enable = 1) an attempt to open RoCE QPs will fail. RoCE can be disabled only after all RoCE QPs are destroyed. Otherwise, RoCE disable command (nic.roce_enable = 0) will fail. Please note that RoCE Address Table can be configured even when RoCE is disabled.	

Table 30 - NIC_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
04h	31	arm_change_event	Set by vport_group_manager to arm generating Nic_Vport_Change_Event when one of the fields with event_on_XXX ==1 changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
	4	event_on_mtu	Event on mtu field change. Set by vport_group_manager to ask for event if one of the mtu field changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
	3	event_on_promisc_change	Event on promisc field change. Set by vport_group_manager to ask for event if one of the promisc fields changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
	2	event_on_vlan_change	Event on vlan list change. Set by vport_group_manager to ask for event if vlan list changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
	1	event_on_multicast_address_change	Event on multicast list change. Set by vport_group_manager to ask for event if multicast list changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
	0	event_on_unicast_address_change	Event on unicast list change. Set by vport_group_manager to ask for event if multicast list changed. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.	
08h	19:16	affiliation_criteria	Specifies the criteria for affiliating traffic hitting the NIC vport with another VHCA. See Section 14.4, “Multi-port vHCA,” on page 845. 0x0: NO_AFFILIATION 0x1: RDMA_TRAFFIC – affiliate ROCE traffic to the vhca specified by affiliated_vhca_id. Note that vport roce_en field must be set when affiliating the NIC vport with this criteria.	
	15:0	affiliated_vhca_id	Specifies the VHCA that the NIC vport is affiliated with.	
24h	15:0	mtu	MTU size required for Vport. In query, 0 means MTU of Vport equal to external physical port. [Internal] 0 come because of compatability issue of old FW. New driver mustn't set 0.	

Table 30 - NIC_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
28h-2Ch	64	system_image_guid	System image guid	
30h-34h	64	port_guid	Port GUID	
38h-3Ch	64	node_guid	Node GUID	
40h	31:16	max_qp_retry_limit	See max_qp_retry_period description.	
	15:0	max_qp_retry_period	If both max_qp_retry_limit>0 and max_qp_retry_period>0, then for all Reliable Connection RDMA QPs (RC/XRC/DCI) associated with this vport: If max_qp_retry_limit retransmissions occur within max_qp_retry_period seconds the QP will move to error state and generate an error CQE with syndrome=0x15 - Transport Retry Counter Exceeded. Supported only if HCA_CAP.max_qp_retry_freq==1.	
68h	31:16	qkey_violation_counter	IB spec Qkey violation counter.	
F0h	31	promisc_uc	When set, request to allow any unicast address on vlan list.	
	30	promisc_mc	When set, request to allow any multicast address on vlan list.	
	29	promisc_all	When set, request to allow all traffic.	
	26:24	allowed_list_type	Determine the allowed list type 0: current_uc_mac_address 1: current_mc_mac_address 2: vlan_list	
	11:0	allowed_list_size	Size of the addresses list. The list type is defined by allowed_list_type. For allowed_list_type==current_uc_mac_address, allowed_list_size must be <=HCA_CAP.log_max_current_uc_list. For allowed_list_type==current_mc_mac_address, allowed_list_size must be <=HCA_CAP.log_max_current_mc_list For allowed_list_type==current_uc_mac_address, allowed_list_size must be <=HCA_CAP.log_max_vlan_list	

Table 30 - NIC_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
F4h-F8h	64	permanent_address (See Table 33, “MAC Address Layout,” on page 293)	Permanent address. MAC address format. See Table 33, “MAC Address Layout,” on page 293.	
FCh	31:0	sw_network_metadata	Software Network metadata. Uses by the e-Switch manager software to pass tunnel information to its vports on the e-Switch. 0 means reserved. Supported for set only when HCA_CAP.sw_network_metadata==1.	
100h-...	64	current_uc_mac_address[...]/current_mc_mac_address[...]/vlan[...]	List of the allowed addresses on a nic_vport. The size of the list is determined by allowed_list_size, and the type of the list is determined by allowed_list_type. For the format of uc_mac_address/mc_mac_address: See Table 33, “MAC Address Layout,” on page 293. For the format of vlan, See Table 31, “Vlan Layout,” on page 292.	

Table 31 - Vlan Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				vlan								00h				
																											04h					

Table 32 - Vlan Field Descriptions

Offset	Bits	Name	Description	Access
00h	11:0	vlan	vlan	

Table 33 - MAC Address Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																mac_addr[47:32]												00h				
mac_addr[31:0]																																04h

Table 34 - MAC Address Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	mac_addr[47:32]	Upper bits of mac address.	
04h	31:0	mac_addr[31:0]	Lower bits of mac address.	

The Table below indicates which vnic is responsible to modify fields in NIC VPORT CONTEXT.

NIC_VPORT_CONTEXT attributes will return back to default when the ‘attribute owner’ undergoes reset.

Table 35 - MODIFY_NIC_VPORT_CONTEXT Bitmask

Bit	Field	CAPABILITY	Modified by
16	sw_network_metadata	HCA_CAP.sw_network_metadata	e-Switch manager only for other vNICs (including its vport) (HCA_CAP.eswitch_manager)
15	system_image_guid	HCA_CAP.system_image_guid_modifiable	HCA_CAP.vport_group_manager vport_group_manager for others that are not vport group manager. ECPF allowed to modify it for other vport group managers vnics (not for itself)
None	qkey_violation_counter	RO	NA
14	max_qp_retry_limit/max_qp_retry_period	HCA_CAP.max_qp_retry_freq	HCA_CAP.vport_group_manager
13	affiliated_vhca_id	HCA_CAP.affiliate_nic_vport_criteria	Vnic to itself
13	affiliation_criteria	HCA_CAP.affiliate_nic_vport_criteria	Vnic to itself

Table 35 - MODIFY_NIC_VPORT_CONTEXT Bitmask

Bit	Field	CAPABILITY	Modified by
12	multi_prio_sq	HCA_CAP.multi_prio_sq	Vnic to itself
11	disable_uc_local_lb	disable_local_lb_uc	Vnic to itself
10	disable_mc_local_lb	HCA_CAP.disable_local_lb_mc	Vnic to itself
9	node_guid	HCA_CAP.vport_group_manager	vport_group_manager for otherst-hat are not vport group manager. ECPF allowed to modify it for other vport group managers(vnics (not for itself)
8	port_guid	HCA_CAP.vport_group_manager	vport_group_manager for otherst-hat are not vport group manager. ECPF allowed to modify it for other vport group managers(vnics (not for itself)
7	min_wqe_inline_mode	HCA_CAP.eswitch_manager	eswitch manager only for other vnics (not for it self) (HCA_CAP.eswitch_manager)
6	mtu	None	Vnic to itself
5	arm_change_event	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_mtu	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_promisc_change	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_vlan_change	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_mc_address_change	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_uc_address_change	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
5	event_on_uc_address_change	HCA_CAP.nic_vport_change_event	eswitch manager (HCA_CAP.eswitch_manager)
4	promisc_uc	None	Vnic to itself
4	promisc_mc	None	Vnic to itself
4	promisc_all	None	Vnic to itself

Table 35 - MODIFY_NIC_VPORT_CONTEXT Bitmask

Bit	Field	CAPABILITY	Modified by
3	permanent_address	HCA - CAP.vport_group_manager	vport_group_manager for others that are not vport group manager. ECPF allowed to modify it for other vport group managers vnics (not for itself)
2	allowed_list_type	None	Vnic to itself
2	allowed_list_size	None	Vnic to itself
2	current_uc_mac_address[...]/ current_mc_mac_address[...]/ vlan[...]	None	Vnic to itself
1	roce_en	None	Vnic to itself

8.8 HCA_Vport Context - HCA Virtual Port Context

Note that this context is supported only when HCA_CAP.ib_virt==1. See Table 1571, “HCA Capabilities Layout,” on page 1671.

Table 36 presents HCA Vport context structure.

Table 36 - HCA_Vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
field_select																																00h
																																04h-1Ch
sm_virt_aware	has_smi	has_raw	grh_required		min_wqe_inline_mode											port_physical_state	vport_state_policy	port_state	vport_state	20h												
																																24h
system_image_guid																																28h-2Ch
port_guid																																30h-34h
node_guid																																38h-3Ch

Table 36 - HCA_Vport Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
cap_mask1																																40h
cap_mask1_field_select																																44h
cap_mask2																																48h
cap_mask2_field_select																																4Ch
																																50h-5Ch
lid																	init_type_re- ply	lmc		subnet_timeout								60h				
sm_lid																sm_sl																64h
qkey_violation_counter																pkey_violation_counter																68h
																																6Ch-1FCh

Table 37 - HCA_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	field_select	A bitmask to each field in the context which informs the following - In QUERY if fields are RO or RW - In MODIFY indicates which fields to modify Each bit in field_select has this information for a specific field in context. Value 0 - means RO. Value 1 - means RW. ‘ field_select bitmask Bit 0: port_guid Bit 1: node_guid Bit 2: vport_state_policy Bit 3: min_wqe_inline_mode - used only by vport_group_manager to configure other ports (not its own vport) Bit 4: grh_required - Allows PFs to determine the VF VPort type. Bit 5: system_image_guid - allows vport group manager to modify the system_image_guid of its VFs	

Table 37 - HCA_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
20h	31	sm_virt_aware	SM is virtualization aware.	
	30	has_smi	If set, this port can be used to send and receive SMPs. Otherwise, SMP traffic should be silently dropped.	
	29	has_raw	If set, this port can send and receive Raw Ethernet and Raw IPv6 packets. Otherwise, Raw traffic should be silently dropped.	
	28	grh_required	When set, the associated port type is GID-based, mandating a GRH for all traffic. Otherwise, the port type is LID-based.	
	26:24	min_wqe_inline_mode	Sets the minimal allowed inline mode for NIC Vport SQs. Each SQ has inline mode that defines the minimal required inline headers in Eth Segment of the SQ's WQEs. Note: SQ.min_wqe_inline_mode must be >= Nic_vport.min_sq_wqe_inline_mode. Must be configured before the VF is loaded. For more details - See Section 8.9.4.1.5, "Eth Segment and Padding Segment," on page 318. Valid only when HCA_CAP.wqe_inline_mode == 1	
	15:12	port_physical_state	The physical link state as specified in IB spec (PortInfoPortPhysicalState) 1: Sleep 2: Polling 3: Disabled 4: PortConfigurationTraining 5: LinkUp 6: LinkErrorRecovery 7: PhyTest	
	11:8	vport_state_policy	vport state policy 0: Down 1: Up 2: Follow	
	7:4	port_state	The logical port state of the physical port as specified in IB spec (PortInfo.PortState) 1: down 2: init 3: arm 4: active	
	3:0	vport_state	Virtual port state 1: down 2: init 3: arm 4: active	

Table 37 - HCA_Vport Context Field Descriptions

Offset	Bits	Name	Description	Access
28h-2Ch	64	system_image_guid	System image GUID	
30h-34h	64	port_guid	Port GUID	
38h-3Ch	64	node_guid	Node GUID	
40h	31:0	cap_mask1	IB spec CapabilityMask1	
44h	31:0	cap_mask1_field_select	A bitmask to each bit in the cap_mask which informs: - In QUERY if the bit is RO or RW - In MODIFY indicates which to modify Value 0 - means RO Value 1 - means RW	
48h	31:0	cap_mask2	IB spec CapabilityMask2	
4Ch	31:0	cap_mask2_field_select	A bitmask to each bit in the cap_mask which informs: - In QUERY if the bit is RO or RW - In MODIFY indicates which to modify. Value 0 - means RO Value 1 - means RW	
60h	31:16	lid	IB spec - the base LID of the port.	
	11:8	init_type_reply	IB spec InitTypeReply.	
	7:5	lmc	IB spec LID mask count for multipath support.	
	4:0	subnet_timeout	IB spec subnet timeout.	
64h	31:16	sm_lid	The LID of the SM managing this port.	
	15:12	sm_sl	The administrative SL of the SM managing this port.	
68h	31:16	qkey_violation_counter	IB spec Qkey violation counter.	
	15:0	pkey_violation_counter	IB spec Pkey violation counter.	

8.9 Work Queues

Work Request is posted to the HCA by writing a list of one or more Work Queue Elements (WQE) to the WQ and ringing the DoorBell, notifying the HCA that request has been posted.

This section describes the structure and management of WQs and the WQE format.

HCA WQ is an object containing the following entities:

1. **QP/SQ/RQ Context** - Contains control information required by the device to execute I/O operations on that Context. These contexts are configured by SW at creation time.
2. **Work Queue Buffer** - A virtually-contiguous memory buffer allocated when creating the QP/SQ/RQ:
 - **Send Queue** - A virtually-contiguous circular buffer accessible by user-level software and used to post send requests. Maximum supported SQ size is retrieved by the QUERY_HCA_CAP command (*log_max_qp_sz*), see Table 1571, “HCA Capabilities Layout,” on page 1671.
 - **Receive Queue** - A virtually-contiguous circular buffer accessible by user-level software and used to post receive requests. Maximum supported RQ size is retrieved by the QUERY_HCA_CAP command (*log_max_qp_sz*), see Table 1571, “HCA Capabilities Layout,” on page 1671.
3. **DoorBell Record** - A structure containing information of most recently-posted Work Request.

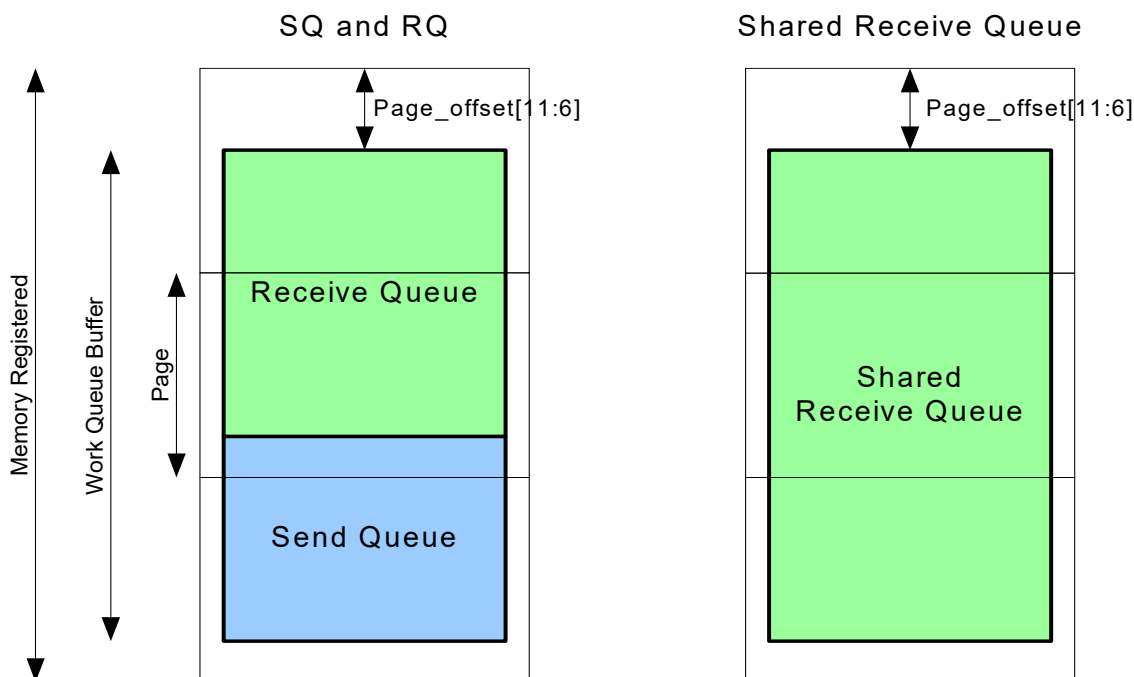
8.9.1 Work Queues Structure and Access

The device WQ is a virtually-contiguous memory buffer used by SW to post I/O requests (WQEs) for HCA execution. A WQ is comprised of WQE Basic Blocks (WQEBBs); WQE size is modulo of WQEBB size.

Each WQ buffer contains the Send WQ and Receive WQ adjacently. The RQ resides in the beginning of the buffer; Figure 20 illustrates the structure of Work Queue Buffer.

The buffer must be aligned on WQEBB/Stride (the larger of the two); RQ must be aligned to 64B even if RQ stride is smaller than 64B. RQ can be of zero size. The offset of WQE in the first page (*page_offset*) is delivered to the device while configuring the QP/SQ/RQ.

Figure 20: Work Queue Buffer Structure



What drives this requirement? We will not need it for DCT QPs that are associated with an SRQ are considered to not have an RQ. In this case all parameters that describe the RQ are reserved.

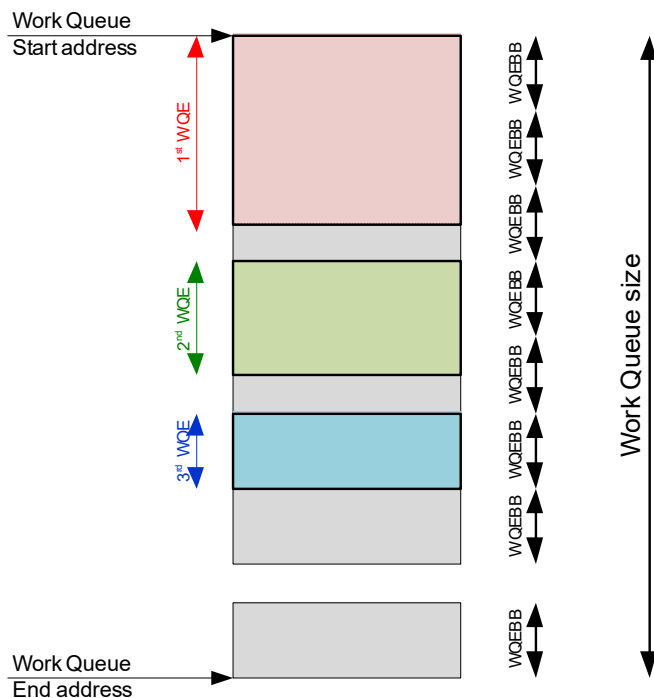
8.9.1.1 Send Queue

The size of Send WQ is specified in units of Work Queue Basic Blocks (WQEBBs). The size of a WQEBB is 64 bytes. The size (or depth) of the SQ is a power-of-two number of WQEBBs, and therefore the queue itself is a power-of-two size (in bytes). The maximum SQ size in WQEs can be retrieved through the QUERY_HCA_CAP command ($2^{\log\_max\_qp\_sz}$).

SQ is organized as a circular buffer containing Send WQEs. Each posted WQE can occupy one or more basic blocks. This enables using large WQEs without wasting memory on huge WQs.

Note that the value of **log_sq_size** does not necessarily equal the number of WQEs that can be posted to the SQ. Rather, it reflects the maximum number in case where the size of each posted WQE is smaller than or equal to WQEBB. Thus, the number of WQEs that can be posted may vary depending on the actual posted WQE sizes. The maximum WQE size can be retrieved by QUERY_HCA_CAP command (and **max_wqe_sz_sq**). Figure 21 illustrates the Send WQ with three WQEs of different sizes posted.

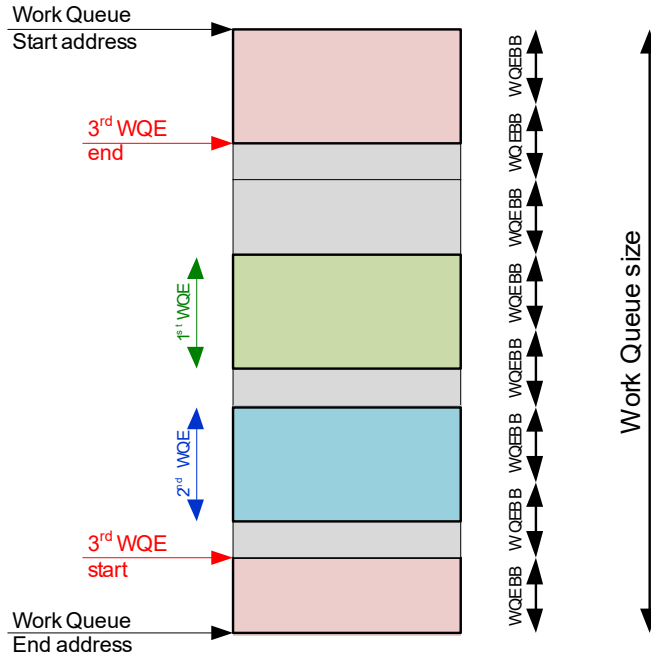
Figure 21: Send Work Queue With 3 WQEs Posted



SQ WQEs are identified by *sq_wqebb_counter*, which starts at zero and advances with each WQE post by number of WQEBBs this WQE occupies and is stored in *Send Doorbell Record*. *sq_wqebb_counter* wraps around 0xFFFF.

WQE can wrap around the Send WQ if it starts close to the edge of the WQ and does not fit into the space left. This phenomena is illustrated in Figure 22.

Figure 22: WQE Wrap-around in WQ Buffer



8.9.1.2 Receive Queue

RQ is organized as a circular buffer containing Receive WQEs. Receive WQEs are placed at fixed stride in the Receive WQ buffer and are executed in the order of their appearance in the buffer. The stride is specified in the QP/WQ in a multiple of 16-byte chunks; the number of these chunks must be a power of two. The maximum RQ size in strides can be retrieved through the QUERY_HCA_CAP command ($2^{\log\_max\_qp\_sz}$). Maximum WQE size supported is retrieved by QUERY_HCA_CAP command. Can be 512B or smaller.

Each 16-byte chunk of Receive WQE contains a single scatter pointer and byte count. If WQ stride size is larger than the actual WQE, the scatter pointers list can be terminated with scatter pointer specifying a zero value in the *byte_count* field and *L_Key* = QUERY_SPECIAL_CONTEXT.terminate_scatter_list_mkey. This scatter pointer is interpreted by HW as end of the scatter list and is referred thereafter as *Null* scatter pointer. When WQE signature is enabled for the RQ, the first segment in the WQE is signature segment. See Figure 38, “Receive Queue - Signature Entry Format”. For more details on RQ entry signature, see Section 17.2.3, on page 879.

Table 38 - Receive Queue - Signature Entry Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	0											00h

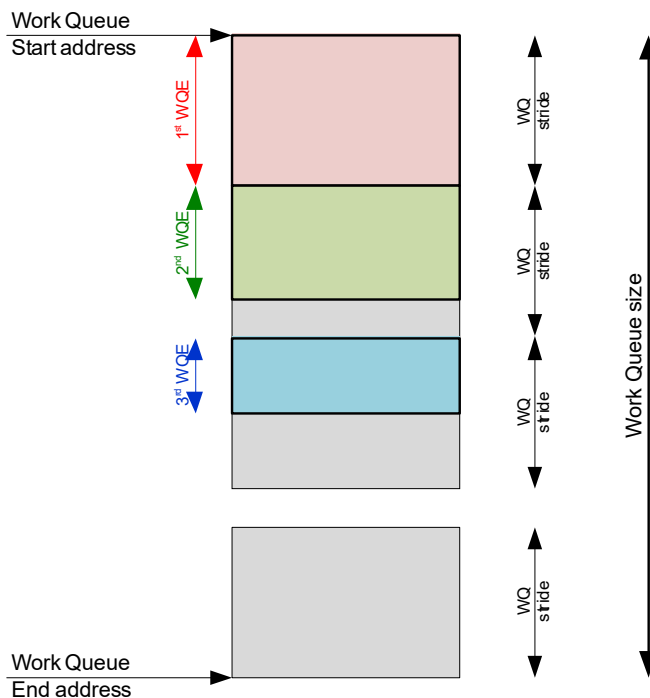
Table 38 - Receive Queue - Signature Entry Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2														04h
signature																												08h					
																												0Ch					

Table 39 - Receive Queue - Signature Entry Fields

Offset	Bits	Name	Description
04h	31:24	signature	WQE signature. The signature covers the entire WQE as well as the <i>QP/RQ number</i> and the <i>WQE index</i> corresponding to that WQE. Should result in 0xff. This feature is enabled per QP/WQ. The feature covers the whole WQE stride including the NULL segments.

Figure 23 illustrates the Receive WQ with three WQEs posted.

Figure 23: Receive Work Queue with 3 WQEs Posted


RQ WQEs are identified by the *rq_wqe_counter*, which starts at 0 and advanced by 1 each time a WQE is posted to the RQ and stored to *Receive Doorbell Record*. *rq_wqe_counter* wraps around 0xFFFF.

8.9.1.3 Striding Receive WQ

Note: This feature is supported for Ethernet when `HCA_CAP.eth_striding_wq==1` and for IB when `HCA_CAP.ib_striding_wq==1`.

Legacy Receive WQs use a message-level buffering scheme - buffers are allocated and consumed per incoming message, where a message in this context is either:

1. A raw Ethernet packet
2. An LRO aggregate
3. An IB transport message.

As the size of incoming messages is (most usually) un-predictable, the size of each receive buffer must be of the maximum possible incoming message size.

This legacy Receive Work Queue scheme has the following two inefficiencies:

- Receive buffers memory utilization:
 - a. Incoming messages that are smaller than the maximum possible message size leave receive buffers memory unutilized. The bigger the difference between the maximum and minimum message size the more severe this in-efficiency becomes.
 - b. For example: an LRO-enabled RWQ with a maximum LRO session size of 64KB requires receive buffers of 64KB. An incoming 64B packet will then consume a 64KB buffer, yielding a poor memory utilization of 0.1%.
 - c. A consequence of this in-efficiency is reduced ability to absorb bursts of small incoming messages which may lead to packet drops.
- Receive WQE fetch per incoming packet:
 - a. A receive WQE must be fetched from host memory (unless already cached on the device) per incoming packet.
 - b. In many scenarios this becomes the performance bottleneck of the device RX processing.

The Striding WQ (STRWQ) is intended to resolve these legacy RWQ in-efficiencies.

It is achieved by replacing the legacy message-level buffering scheme by a byte-level buffering scheme.

With STRWQ, each incoming packet (or an LRO aggregate) consumes a buffer of its size (up to nuances) rather than of the maximum possible incoming message size, thus resolving the memory utilization in-efficiency). Therefore, a single WQE can host multiple incoming messages, as many as its size limitation enables, thus resolving the WQE fetch per message in-efficiency.

8.9.1.3.1 STRWQ WQEs

An STRWQ is comprised of WQEs just like legacy WQs, however, with STRWQ each WQE can host multiple incoming messages each generating its own CQE. A single packet (or LRO aggregate) resides in a single WQE and cannot cross WQEs.

To leverage the STRWQ capability of reduced WQE fetch overhead, SW should use big STRWQ WQE buffers. Thus, it is anticipated that an STRWQ will be comprised of fewer and bigger WQEs compared to legacy RWQ.

8.9.1.3.2 STRWQ Strides

The receive buffers of an STRWQ are segmented into fixed size strides. The STRWQ stride size is determined by the `WQ.single_stride_log_num_of_bytes` field. An incoming message always starts on a stride boundary. The STRWQ stride is intended to align the start of incoming messages to the system cache line for avoiding an in-efficient Read-Modify-Write operation on the host memory caused by partial cache line writes. Thus, the STRWQ stride size should not be smaller than the system cache line size.

8.9.1.3.3 STRWQ Completion

As each STRWQ WQE can host multiple incoming messages, the legacy 1:1 WQE-CQE relationship is no longer valid. Thus, to avoid CQ overrun, the size of a CQ associated with an STRWQ must accommodate room for the maximum number of packets that the STRWQ can store. That is, the CQ size in CQEs, per associated STRWQ, must be equal to or greater than the STRWQ size divided by the minimum packet size rounded up to the WQ stride size.

As each WQE can host multiple CQEs, each CQE consumes just a portion of a WQE buffer. That portion may be greater than the completed packet size and is in STRWQ stride granularity. Therefore, the CQE must specify:

1. The WQE stride index of the packet start:
 - For CQE of a Raw Ethernet RQ: `CQE.wqe_counter[15:0]` (and then the `CQE.wqe_id` field specifies the WQE)
 - For CQE of IB transport QP: `CQE.ib_stride_index`
2. The number of consumed strides:
 - `CQE.byte_cnt[29:16]` (and `byte_cnt[15:0]` specifies the actual byte count of the packet)

For Raw Ethernet STRWQ, to align the headers following the Ethernet header to a 4-bytes DWORD, SW may set the `WQ.two_byte_shift_en` bit. When set, the device will scatter the packet starting at an offset of two bytes from the start of a stride and pad the first two bytes of the stride in order to avoid writing a partial cache line which results with an in-efficient Read-Modify-Write operation on the host memory.

For IPoIB, SW should not set this bit, since the IB link layer headers are already 4 bytes aligned.

8.9.1.3.4 IB Transport STRWQ

IB transport QPs, associated with a legacy RWQ, generate a CQE per incoming IB transport message. This is achieved by allocating a whole WQE for the destination QP upon first-of-a-message packet (as specified by the IB packet BTH opcode field) arrival.

With STRWQ, IB transport QPs associated with an STRWQ generate a CQE per packet, not per message. Different packets of the same IB transport message consume STRWQ buffers independently of each other. Thus, one packet of a message may consume a portion of one WQE while another packet of the same message may consume any other portion of any WQE.

As each packet generates a CQE, a new bit is added to the CQE - `CQE.byte_cnt[30]`, specifying whether the IB transport packet is last/only of a message. This allows SW to aggregate the scattered IB message packets per QP and pass them as a completed message to the IB verbs consumer application.

8.9.1.3.5 STRWQ WQE Data Buffer Size

The WQE data buffer size (the total size of its data segments) of all WQEs of an STRWQ must be equal.

That size is determined by the $WQ.single\_wqe\_log\_num\_of\_strides * WQ.single\_stride\_log\_num\_of\_bytes$.

When `HCA_CAP.ext_stride_num_range=0`, the minimum number of WQE strides is 512.

When `HCA_CAP.ext_stride_num_range=1`, smaller values are supported.

8.9.1.3.6 WQE Ownership

A WQE moves from device ownership to SW ownership when all its strides are consumed. To conclude that, SW needs to sum the consumed strides per WQE. It should hold a “consumed_strides” counter per WQE, initialize it to zero before posting the WQE to the device and increment it upon each CQE by `CQE.byte_cnt[29:16]`. Then, when this counter reaches the WQE size in strides, the device is done using the WQE and it moves to SW ownership.

The device may also release STRWQ WQE strides by generating filler CQEs. A filler CQE is indicated by `CQE.byte_cnt[31]=1` and it is not associated with any incoming packet. Thus its `byte_cnt[15:0]` field is not valid. The only intention of a filler CQE is for SW to increment its WQE “consumed_strides” counter.

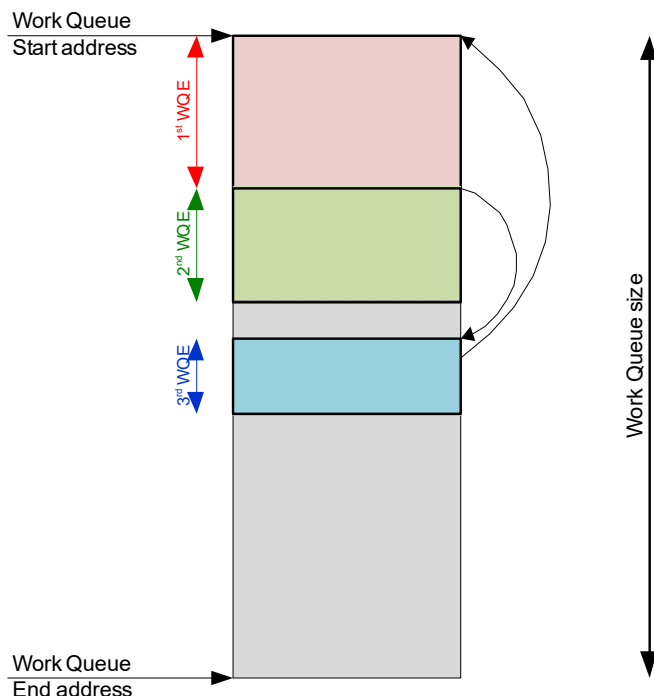
8.9.1.4 Shared Receive Queue

SRQ is organized as a buffer containing linked-list of Receive WQEs. Like on RQ, WQEs on SRQ are placed at fixed stride in the buffer. However - unlike RQ - the order to WQEs execution in SRQ is controlled by link-list. Each WQE on SRQ contains a pointer to the next WQE to be executed. This pointer is located in the first 16-byte chunk of the WQE, which composes the *next/control* segment of the WQE.

Each subsequent 16-byte chunk of SRQ WQE contains a single scatter pointer and byte count. SRQ WQE is terminated same way as RQ WQE (either WQE stride exhausted or *Null* scatter pointer), see “Posting Work Request on Shared Receive Queue” on page 310. Maximum WQE size is retrieved by `QUERY_HCA_CAP` command.

Figure 24 illustrates the SRQ with three WQEs posted.

Figure 24: Shared Receive Queue With 3 WQEs Posted



SRQ WQEs are addressed by the *wqe_index* (only for the purpose of reporting completions). The *wqe_index* refers to the index, in WQE stride, of the WQE posted/completed.

The maximum SRQ size in strides can be retrieved through the QUERY_HCA_CAP command ($2^{\log\_max\_srq\_sz}$).

8.9.1.4.1 Shared Receive Queue - Performance Considerations

For best performance WQEs should be posted in sequential order in the WQ. During the device operations, SRQ Work Requests can be completed out-of-order, thereby creating “holes” in the SRQ. While posting new WQEs to the SRQ, it is advised to minimize discontinuity of the WQEs:

1. While posting WQEs on the SRQ, use the WQE index located contiguously to the last WQE posted on the SRQ
2. If a contiguous WQE index is not available, locate the largest contiguous chunk of available WQEs and post the new WQEs sequentially at this contiguous chunk.

8.9.1.5 [EMC] Dual Write Queue

The device enables the posting of a single RDMA write Work Request on a send Work Queue and have this request executed on two different Send Queues, one that is used when posting the request and another one with an adjacent QP number. Such an operation will result in two messages sent by the device. Each one will have a distinctly different address in the RDMA header and will send the same data to two destinations. The data will be read once on PCI and sent twice

on the network. Completions will be reported separately for each of the two Send Queues. Use of the dual write feature may significantly reduce PCI bandwidth use.

SW should wait for both completions to make sure that the data has arrived at both destinations before assuming the transfer was good.

Dual-write effectively fuses two Send Queues at their creation, and only “parent” Send Queue should be used to post Work Requests. Fused Send Queues must comply with the following requirements:

1. Both Transport Contexts must use same network port of the device.
2. Both address paths must use same network MTU.
3. Both address paths must use same SL (to assure they are mapped to same VL on the way out).
4. Both Send Queues must belong to same Protection Domain.
5. Only local addressing - no GRH in dual write packets.
6. Same PKey for both QPs.
7. Loopback is not supported.
8. Responder side of Dual Write queues can't be used.
9. Strong Ordering is not supported
10. Pending (posted and not completed) WQEs must contain less than $2^{23} * \text{MTU}$ bytes of data in the air at any given time (WQE data should be rounded up to nearest MTU). It is SW's responsibility to enforce this constraint - HW does not have a mechanism which can detect this violation.
11. With rPSV on dual write QP, the MKey to which the result is written should not be “bound to QP”

Retransmission is not supported (retry_count and rnr_retry fields in QP context must be 0) These queues cannot be used by any other operation but for dual write, UMR and PDSV handling WQEs.

8.9.1.5.1 Creation of Dual-write QP

For creating dual write queues, the user should create both the *parent* QP (with even qp-number) and the *child* QP (with an adjacent QP number) using two different CREATE_QP commands. Both will have the dual_write bit set. The user should run MODIFY_QP commands to move both QPs to RTS. The child QP should be moved to RTS before the parent QP.

Both QPs share the work queue. Therefore, they have the same MTT_index that points to the WQ. To handle this (temporary, will be changed in the coming release) we will need to create the child QP first. When creating the parent QP, we need to set the keep_mtt opcode modifier.

To destroy dual-write pair, the parent QP should be destroyed first (with the keep_mtt opcode modifier); then the child QP is destroyed without this modifier.

8.9.2 Doorbell Record

Doorbell records are located in physical memory. The address of DoorBell record is passed to the HW at QP/RQ/SQ creation (see “Queue Pair Context Format” on page 258). The receive and send DoorBell record are consecutive in memory. DB records are aligned on a 4 byte boundary (bits[1:0] of the address must be 0). The DoorBell record denotes the counter (*sq_wqebb_counter* for SQ and

wqe_counter for receive and shared RQs), which reflects the number of WQEs posted to this queue since its creation. Specifically, it points to the first empty WQEBB (SQ) or WQE stride (RQ). The counter wraps around at 0xFFFF for all other queues. Doorbell Record format is shown in Table 40.

Table 40 - Doorbell Record Format

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	11 0	1 0	9	8	7	6	5	4	3	2	1	0	Offset
																rcv_counter											00					
																send_counter											04					

Table 41 - Doorbell Record Fields

Offset	Bits	Name	Description
00h	15:0	rcv_counter	Receive counter
04h	15:0	send_counter	Send counter

Doorbell record should be initiated to 0 upon creation (both rcv and send counters).

[EMC] when `rq_type=no_rqw` and `sq_type=no_sq=1`, user can avoid from creating DoorBell records.

8.9.3 WQE Ownership

Hardware reads and executes WQEs from WQs asynchronously to SW posting new WQEs; each WQE is executed only once, and completion is optionally reported to the CQ associated with WQ. The WQE can be executed by the HCA only if its ownership has been passed to HW.

The WQE ownership is passed to HW by updating the *Doorbell Record*, posting there respective WQE counter (*sq_wqebb_counter* and *rq_wqe_counter* for send and receive DoorBell records respectively).

Once WQE ownership has been passed to the HCA, HW can execute the WQE. Neither WQE nor data buffers associated with this WQE can be modified by SW until WQE ownership has been returned back to the SW. Altering WQE or associated data buffer can have lethal consequences to that WQ. It is guaranteed, however, that operation of other queues will not be affected.

After a WQE has been executed, its ownership is passed to SW by posting CQE to respective CQ.

WQE processing is performed on QPs that are in RTR or RTS state and on RQs and SQs that are in RDY state. To transfer the QP into RTR or RTS states, the driver should use the QP commands as described in “QP Commands” on page 1897. To transfer the RQ or SQ into RDY state, the driver should use the RQ and SQ commands as described in Section 29.14 and Section 29.15. Prior to moving a QP into INIT state, the device driver has to initialize DoorBell counters to zero. Prior to moving RQ or SQ from RST into RDY state, the device driver has to initialize DoorBell counters to zero.

8.9.3.1 Posting a Work Request to Work Queue

In order to post Work Request to WQ, software should execute the following steps:

1. Write WQE to the WQE buffer sequentially to previously-posted WQE (on WQEBB granularity).
2. Update *Doorbell Record* associated with that queue by writing their *sq_wqebb_counter* or *wqe_counter* for send and RQ respectively
3. For send request ring DoorBell by writing to the Doorbell Register field in the UAR associated with that queue. For performance-critical send WQEs DoorBell can be rang by using the *BlueFlame* mechanism (see “Blue Flame” on page 253).

For send requests the third step (DoorBell) *guarantees* that send WQE will be executed. However, HW *may* execute WQE after the second step has been completed. Ringing DoorBell for WQE that has already been executed makes no harm - such a DoorBell is ignored by the HW.

Software can post a list of WQEs together by repeating step 1 for all WQEs, then updating the DoorBell record (step 2) once and ringing one DoorBell (step 3). The DoorBell record *wqebb_counter* should count all WQEs posted and the *wqe_index* in the DoorBell should point to the first wqebb of the last WQE posted.

For receive requests, updating DoorBell record passes receive WQE ownership to HW and receive operation will be executed upon arrival of respective send packet.

8.9.3.2 Posting Work Request on Shared Receive Queue

While posting Work Request to the SRQ, software should take the following steps:

1. Locate a slot in WQ to be used for the WQE that will be posted after the current one being posted and write its pointer to *next/control* segment of the newly-created WQE. Since *next/control* segment of the WQE contains address of next WQE, the location of next WQE to be posted is determined while posting a current WQE.
2. Write WQE (the scatter list, concluding with *Null* scatter pointer if required) to the slot located in step (1) of previously-posted WQE.
3. Update *Doorbell Record* associated with that SRQ writing there *rq_wqe_counter*.

The third step (DoorBell record update) passes receive WQE ownership to HW and receive operation will be executed upon arrival of respective send request.

8.9.4 Work Request (WQE) Formats

The maximum length of a Work Request (WQE) reported in *HCA_CAP.max_wqe_sz_rq* and *max_wqe_sz_sq*, and is specified in octowords (16-byte chunks). WQEs must be aligned on WQEBB/Stride.

8.9.4.1 Send WQE Format

Send WQEs are built from the segments that are listed below. Segment size is a multiple of 16 bytes. Each WQE contains a subset of these segments, depending on the transport service of the QP and the requested operation, as shown in Table 106, “WQE Construction Summary”. For each transport service and operation, segments must be in accordance with Table 106, “WQE Construction Summary” and with the order shown in the list below:

- *Ctrl Segment* - The segment contains some control information for the current WQE. This segment is described in Section 8.9.4.1.1, “Ctrl Segment,” on page 312.
- *MLX Segment* - *The segment contains destination addressing information for InfiniBand MLX transport. This segment is described in Section 8.9.4.1.18, “MLX WQE Format,” on page 343.*
- *DC Transport Segment* - Contains address information required to form a DC Transport message. This segment is described in Section 8.9.4.1.3, “DC Transport Segment,” on page 317.
- *XRC Segment* - Contains the Receive Buffer Queue (SRQN) to be used on remote side. This segment is described in Section 8.9.4.1.4, “XRC Segment,” on page 317.
- *Datagram Segment* – Contains address information required in order to form a datagram message. The unreliable datagram segment is described in Section 8.9.4.1.2, “Datagram Segments,” on page 317.
- *Eth Segment* - Contains packet headers and information for stateless L2, L3, L4 offloading. This segment is described in Section 9.3.4.1.5, “Eth Segment and Padding Segment,” on page 140.
- *Remote Address Segment* – Present only in RDMA or Atomic WQEs. It specifies remote virtual addresses and RKEY, respectively. The length of a remote access is calculated from the scatter/gather list (RDMA-write/RDMA-read, extended atomic) or set to 8 (Atomic). This segment is described in Section 8.9.4.1.11, “Remote Address Segment,” on page 331.
- *Memory Management Segment* - Contains the parameters required for a Memory Management WQEs (UMR). This segment is described in Section 8.9.4.1.6, “User-mode Memory Registration (UMR) WQE Format,” on page 322 and on.
- *Atomic Segment* – Present only in ATOMIC WQEs and specifies swap/add and compare data. This segment is described in Section 8.9.4.1.12, “Atomic/Extended Atomic WQE and Segment,” on page 332. This segment can either contain the actual swap/add/compare data inline or a single pointer to a memory location containing these arguments. An exception to this is 8B atomics that do not have a mask, these must have their data inline in the WQE. The data segment of an ATOMIC WQE should contain a single scatter entry where the atomic response will be written.
- *Signature hand-over Segment* – contains signature hand-over control information. These segments are described in “SET_PSV Segment” on page 327, “CHECK_PSV/rCHECK_PSV Segment” on page 328 and “GET_PSV/RGET_PSV Segment” on page 330.
- *Data Segments* - Contain pointers and a byte count for the scatter/gather list. They can optionally contain data, which will save a memory read access for gather Work Requests. The WQE can contain single or multiple memory pointer segments, single or multiple in-line *Data* segments, or any combination of these.

- Tag Matching Segment - Used to perform list of operations on XRQ matching list. This segment should be added after the WQE control segment. Tag matching segment size is 2DS, regardless of the list of operation. See Section 8.9.4.1.17, “Tag Matching Segment,” on page 342.
- ASO Ctrl - Contains the conditions for Advanced Steering Operation (ASO) ASO context updates, data for the ASO context update and the memory location for placing the old value of the context in case a Read operation is required.
- ASO data - Contains data for the ASO context update. ASO update can optionally contain this segment, if required.
- Flow Update Ctrl Segment - Indicates how a Flow Table Entry should be modified and the new destination of this Flow. Used for a Flow Table Entry in a Flow Table configured for WQE based updates.
- Flow Update Data Segment - Contains the field values defining this Flow and the actions required in case this Flow is matched by a packet. Used for a Flow Table Entry in a Flow Table configured for WQE based updates.
-

8.9.4.1.1 Ctrl Segment

This segment contains control information of the WQE. The interpretation and validity of some fields depends on the transport service deployed on the SQ processing the WQE.

Format of the general control segment is shown in Table 42.

Table 42 - General - Ctrl Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
OPC_MOD								wqe_index														OPCODE										
qp_or_sq																							DS									
signature								<u>dci_stream_channel_id</u>														FM			CE		SE	interrupt				
ctrl_general_id																																

Table 43 specifies fields in the control segment.

Table 43 - General Ctrl Segment Field

Offset	Bits	Name	Description
00h	31:24	OPC_MOD	This is OPCODE modifier. See Table 44, “OPC_MOD Field Encoding,” on page 316.
	23:8	wqe_index	WQEBB number of the first block of this WQE. This number should wrap at 0xffff, regardless of size of the WQ.
00h	7:0	OPCODE	Opcode: OpCode of this WQE. Encodes the type of operation to be executed 0x0: NOP - WQE with this opcode creates a completion, but does nothing else 0x1: SND_INV - Send with Invalidate 0x8: RDMA_Write 0x9: RDMA_Write_with_Immediate 0xA: Send 0xB: Send_with_Immediate 0xE: LSO - Large Send Offload (used also for multi-packet send WQE) 0xF: WAIT 0x10: RDMA_Read 0x11: Atomic_Compare_and_Swap 0x12: Atomic_Fetch_and_Add 0x13: Read_and_Invalidate - unsupported [internal] 0x14: Atomic_Masked_Compare_and_Swap - (Extended Atomic operation) 0x15: Atomic_Masked_Fetch_and_Add - (Extended Atomic operation) 0x16: RECEIVE_EN 0x17: SEND_EN 0x20: SET_PSV 0x21: GET_PSV 0x22: CHECK_PSV 0x23: DUMP 0x25: UMR 0x26: RGET_PSV 0x27: RCHECK_PSV 0x28: Tag_Matching 0x29: ENHANCED_MPSW 0x2C: FLOW_TABLE_ACCESS 0x2D: ACCESS_ASO 0x2F: MMO - memory to memory offload 0x30: LOAD_REMOTE_MICRO_APP 0x31: STORE_REMOTE_MICRO_APP
04h	31:8	qp_or_sq	QP/SQ number this WQE is posted to.
	5:0	DS	WQE size in octowords (16-byte units). DS accounts for all the segments in the WQE as summarized in WQE construction

Table 43 - General Ctrl Segment Field

Offset	Bits	Name	Description
08h	31:24	signature	WQE signature. The signature covers (XOR) entire WQE. Should result in 0xff. This feature is enabled per WQ. See Section 17.2.3, “Work Queues Elements Signature,” on page 879.
	23:8	<u>dc_i_stream_chan- nel_id</u>	<u>DCI Stream Channel ID</u>
	7:5	FM	Fence Mode 0x0: No_Fence 0x1: Initiator_Small_Fence - Wait as long as there are WQEs that are currently locally in execution (doing gather / memop etc.) 0x2: Fence - the WQE will start execution only after all previous Read/Atomic WQEs complete. 0x3: Strong_Ordering - when set, the WQE will be executed only after all previous WQEs have been executed. Can be set for RC WQEs only 0x4: Fence_and_Initiator_Small_Fence
	3:2	CE	Completion and Event mode 0x0: cqe_on_cqe_error - Generate CQE only on WQE completion with error. (note - per IB spec, for completion with error, CQE should be generated) 0x1: cqe_on_first_cqe_error - Generate CQE only on first WQE completion with error (i.e if CQE on previous WQE completed in error, no CQE will be generated) 0x2: cqe_always - Generate CQE on WQE completion (good or bad) 0x3: cqe_and_eqe - Generate CQE and EQE (local Solicited event).
	1	SE	Solicited Event
	0	interrupt	Call FW on this WQE (when executed in SXW)

Table 43 - General Ctrl Segment Field

Offset	Bits	Name	Description
0Ch	31:0	ctrl_general_id	General Identifier according to wqe opcode/opc_mod. If the WQE opcode is RDMA-write with Immediate or send with Immediate, then this field holds the Immediate Data field in the message. If the WQE opcode is send with Invalidate, then this field holds the Invalidation key to be sent. If the WQE opcode is UMR and opc_mod is UMR, then this field holds the UMR MKey ({mkey_index[23:0], tag}) If the WQE opcode is LOAD_REMOTE_MICRO_APP or STORE_REMOTE_MICRO_APP, then this field holds the Remote Micro Application operation code described in Table 811, “Remote Micro Application Codes Summary,” on page 1042. If the WQE opcode is UMR and opc_mod is TLS_TIS_STAT-IC_PARAMS, then this field holds the TIS number in [31:8]. Must be !=0. If the WQE opcode is UMR and opc_mod is TLS_TIR_STAT-IC_PARAMS, then this field holds the TIR number in [31:8]. Must be !=0. If the WQE opcode is Send or Dump of TLS accelerated packets, then this field holds the TIRs number in [31:8]. If WQE opcode is ACCESS_ASO and OPC_MOD is IPSEC this field holds the IPSEC Offload object handle. If WQE Opcode is MMO, then ctrl_general_id=mmo_control[63:32] If the WQE opcode is FLOW_TABLE_ACCESS and opc_mod is UPDATE_FLOW, then this field holds the Flow Group ID.

Table 44 specifies OPC_MOD field encoding.

Table 44 - OPC_MOD Field Encoding

OPCODE	7	OPC_MOD[6:0]						
		6	5	4	3	2	1	0
Send Write Write With Imm	0	For 'add' floating point: Round control 0 - trun- cate Else - reserved	0	0		Operation 0 - No CALC 1 - add int 2 - add floating point 3 - maxloc 4 - and 5 - or 6 - xor 7 - RESERVED		
Send Send&Inv Send w/Imm Write Write w/Imm	0xff denotes vector calc segment is used and no other gather element exist in WQE. See Section 8.9.4.1.13, “Vector Calc Segment,” on page 336							
WAIT	0	0	0	0	0	0 - WAIT on CQ PI else - reserved		
Extended Atomics	0	0	0	0	1	Operand length if (OPC_MOD[2:0] == 7) operand_pointer; else 4*2 ^{OPC_MOD[2:0]} bytes;		
UMR	0x0: UMR 0x1: TLS_TIS_STATIC_PARAMS - Supported only when HCA_CAP.tls==1 0x2: TLS_TIR_STATIC_PARAMS - Supported only when HCA_CAP.tls==1							
GET_PSV	0x0: GET_PSV 0x1: TLS_TIS_PROGRESS_PARAMS - Supported only when HCA_CAP.tls==1 0x2: TLS_TIR_PROGRESS_PARAMS- Supported only when HCA_CAP.tls==1 0x3: NVMEOTCP_TIS_PROGRESS_PARAMS - Supported only when HCA_CAP.nvmeotcp==1 0x4: NVMEOTCP_TIR_PROGRESS_PARAMS - Supported only when HCA_CAP.nvmeotcp==1							
SET_PSV	0x0: SET_PSV 0x1: TLS_TIS_PROGRESS_PARAMS - Supported only when HCA_CAP.tls==1 0x2: TLS_TIR_PROGRESS_PARAMS - Supported only when HCA_CAP.tls==1 0x3: NVMEOTCP_TIS_PROGRESS_PARAMS - Supported only when HCA_CAP.nvmeotcp==1 0x4: NVMEOTCP_TIR_PROGRESS_PARAMS - Supported only when HCA_CAP.nvmeotcp==1							
MMO	0x4: REGEXP							
ACCESS_AS O	0x0: IPSEC 0x1: Connection Tracking 0x2; Policer 0x3: Race avoidance 0x4: Hit First							

Table 44 - OPC_MOD Field Encoding (Continued)

OPCODE	7	OPC_MOD[6:0]						
		6	5	4	3	2	1	0
FLOW_TAB LE_AC- CESS	0x0: UPDATE_FLOW 0x1: UPDATE_MODIFY_LIST							

8.9.4.1.2 Datagram Segments

See Table 109, “UD Address Vector Layout,” on page 362.

Per the *InfiniBand Architecture Specification*, if **Q_Key** bit 31 (MSB) is set, the **Q_Key** transmitted in the outgoing packets will be taken from the QPC. Otherwise, the **Q_Key** transmitted in the outgoing packets is the WQE **Q_Key**.

8.9.4.1.3 DC Transport Segment

DC transport segments includes only address vector. See Table 109, "UD Address Vector Layout".

If **DCA_Key** bit 63 (MSB) is set, the **DCA_Key** transmitted in the outgoing packets will be taken from the QPC. Otherwise, the **DCA_Key** transmitted in the outgoing packets is the WQE **DCA_Key**.

8.9.4.1.4 XRC Segment

XRC field specifies the Receive Buffer Queue (SRQN) to be used on remote side. This field goes to the XRCETH. This segment is used for XRC transport service queues only. Format of XRC segment is shown in Table 45.

Table 45 - XRC Segment Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								XRC																				00h				
reserved																															04h-0Ch	

Table 46 - XRC Segment Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:0	XRC		
04h-0Ch	128	reserved		

8.9.4.1.5 Eth Segment and Padding Segment

This segment contains stateless offloading control and inlined Ethernet packet headers. This segment is used for Ethernet SQs as well as for SQs that generate IPoIB or EoIB traffic. When the Eth header is used on Ethernet SQs, it is used as is. When used for IPoIB or EoIB, the Eth segment is preceded by a padding segment, see Table 49, “Padding Segment Format,” on page 322.

The Eth segment contains the headers of the packets inlined within the segment. Headers that are inlined should be excluded from the data segments that follow the Eth segment. The begging of the inline header depends on the protocol offloaded as follows:

- For standard Ethernet - the packet inlined headers start from the Ethernet header.
- For IPoIB packets - the packet inlined headers start from the IPoIB encapsulation header.
- For EoIB packets - the packet inlined headers start from the EoIB encapsulation header.

The minimal required inline headers depend on HCA_CAP.wqe_inline_mode. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.

<TBD> need to review if we want to always inline everything (simple) or not

The format of the Eth segment is shown in Table 47.

Table 47 - Eth Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset																													
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h																													
swp_outer_l4_offset								swp_outer_l3_offset								swp_inner_l4_offset								swp_inner_l3_offset																																				
14cs		13cs		14cs inner		trailer_alignment		swp_outer_l4_type		swp_outer_l3_type		swp_inner_l4_type		swp_inner_l3_type		mss																																												
																																		04h																										
flow_table_metadata																																08h																												
insert_vlan		insert_trailer		trailer_header_association		inline_header_size								inline_headers																		0Ch																												
inline_headers_cont																																10h...																												

Table 48 specifies fields in the Eth Segment.

Table 48 - Eth Segment Fields

Offset	Bits	Name	Description
00h	31:24	swp_outer_l4_offset	Offset of the outer L4 header from start of frame in words (2 Bytes). 0 implies HW parsing. Must not exceed 384 bytes. Supported only when sq.allow_swp is set. See Table 138, “SQ Context Format,” on page 404.
	23:16	swp_outer_l3_offset	Offset of the outer L3 header from start of frame in words (2 Bytes). 0 implies HW parsing. Must not exceed 384 bytes. Supported only when sq.allow_swp is set. See Table 138, “SQ Context Format,” on page 404.
	15:8	swp_inner_l4_offset	Offset of the inner L4 header from start of frame in words (2 Bytes). 0 implies HW parsing. Must not exceed 384 bytes. Supported only when sq.allow_swp is set. See Table 138, “SQ Context Format,” on page 404.
	7:0	swp_inner_l3_offset	Offset of the inner L3 header from start of frame in words (2 Bytes). 0 implies HW parsing. Must not exceed 384 bytes. Supported only when sq.allow_swp is set. See Table 138, “SQ Context Format,” on page 404.

Table 48 - Eth Segment Fields

Offset	Bits	Name	Description
04h	31	l4cs	When set - the device calculates checksum on the L4 header (TCP/UDP) that follows the IP header. (In case of tunneling, it is the one which follows the outer IP header) For non TCP/UDP packets, the device ignores this bit. When l4cs_inner is set must be disabled.
	30	l3cs	When set - the device calculates checksum on the L3 header (IPv4). (In case of tunneling, it is the checksum of the outer L3 header). For non IPv4 packets, the device ignores this bit.
	29	l4cs_inner	When set - the device calculates checksum on the tunneled packet L4 header (TCP/UDP) Reserved for non-tunneling packets. Reserved for tunneled non TCP/UDP packets. When set l4cs must be disabled.
	28	l3cs_inner	When set - the device calculates checksum on the encapsulated packet L3 header (IPv4) Reserved for non encapsulated packets. Reserved for encapsulated non IPv4 packets. When set L4cs must be disabled.
	25:24	trailer_alignment	[Internal] This field is internal only due to HW bug Packet alignment required after inserting the trailer 0x0: NO_ALIGNMENT 0x1: WORD_ALIGNMENT 0x2: DWORD_ALIGNMENT other values are reserved
	21	swp_outer_l4_type	Software parsing L4 header type. 0x0: TCP 0x1: UDP
	20	swp_outer_l3_type	Software parsing L3 header type. 0x0: IPV4 0x1: IPV6
	17	swp_inner_l4_type	Software parsing inner L4 header type. 0x0: TCP 0x1: UDP
	16	swp_inner_l3_type	Software parsing inner L3 header type. 0x0: IPV4 0x1: IPV6
	13:0	mss	Maximum Segment Size - For LSO WQEs - the number of bytes in the TCP payload to be transmitted in each packet Must be 0 on non LSO WQEs.

Table 48 - Eth Segment Fields

Offset	Bits	Name	Description
08h	31:0	flow_table_metadata	This field is carried into transmit packet flow tables in meta-data_reg_a.
0Ch	31	insert_vlan	If set, the device inserts the VLAN header provided in the WQE (as described in the following sentences) into the packet right after the MAC header (i.e the first ethertype will be of the inserted VLAN). insert_vlan cannot be set together with inline_headers.
	30	insert_trailer	When set, a trailer will be added to the packet. The trailer is taken from the first data segment of the WQE, which must be inline. Trailer will be inserted to each packet in case of LSO. Supported only when HCA_CAP.insert_trailer==1.
	28:26	trailer_header_association	Indicates which header this trailer is associated with. the associated header should be updated to include the trailer. Supported only when insert_trailer==1. 0x0: NO_ASSOCIATION - no header will be updated 0x1: OUTER_IP 0x2: OUTER_L4 0x3: INNER_IP 0x4: INNER_L4 other values are reserved
	25:16	inline_header_size	If insert_vlan==0: Length of the inlined packet headers. If insert_vlan==1: Inline_header_size[0:0] indicates the vlan header type: 0x0: C_VLAN - (customer tag, ethertype = 0x8100) 0x1: S_VLAN - (service tag, ethertype = 0x88a8)
	15:0	inline_headers	If insert_vlan==0: Beginning of the inlined packet headers. (description in inline_headers cont'd). If insert_vlan==1: Vlan header {PCP(3 bits), DEI(1 bit), VID(12 bits)}
10h- ...	31:0	inline_headers_cont	The continuation of the inline headers. Can span multiple octowords up to the maximal WQE size. The last octoword of inline packet header can be partially populated. The rest of the bits are reserved.

[internal] The current format exposes only HW parsing

The format of the Padding segment is shown in Table 49:

Table 49 - Padding Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-0Ch
reserved																																

Table 50 specifies fields in the Padding Segment:

Table 50 - Padding Segment Fields

Offset	Bits	Name	Description
00h-0Ch	128	reserved	

8.9.4.1.6 User-mode Memory Registration (UMR) WQE Format

Format of UMR WQE is shown in Table 51.

Table 51 - UMR Work Request Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00-0C
OPCODE = UMR																																10-3C
UMR control Segment (See Table 52, “UMR Control Segment Layout,” on page 323)																																40-7C
MKeyCtx segment (See Table 362, “MKey Context Format,” on page 629)																																
inline KLMs/MTTs/RepeatedBlock (when translations_octword_size!=0 and inline=1) (See Table 56, “UMR Memory Buffer (KLM) Description Argument Format,” on page 326) (See Table 365, “Memory Translation Table (MTT) Entry Layout,” on page 634) (See Table 706, “Repeated Block Format,” on page 919) inline BSFs (whenbsf_octword_size!=0 and inline=1 and (See Table 695, “Byte Stream Format (BSF) Layout,” on page 902) UMR Pointer to KLMs / MTTs/RepeatedBlock and BSFs location (when inline=0) (See Table 54, “UMR Pointer Description Argument Format,” on page 326)																																

Table 52 - UMR Control Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
inline_field			translation_offset_en	check_qpn																											0h	
translation_octword_actual_size																bsf_octword_actual_size_or_translation_offset_15_0												4h				
mkey_ctx_mask																																08h-0Ch
				translation_offset_42_16																												10h
																																14h-2Ch

UMR control segment fields descriptions are shown in Table 53.

Table 53 - UMR Control Segment Fields

Offset	Bits	Name	Description
0h	31	inline_field	When set, BSFs and KLMs/MTTs/Repeated Block are inline in the UMR WQE, else there is a UMR pointer to these structs If no KLMs/MTTs/RepeatedBlock and BSFs, this bit should be set.
	30:29	check_free	0x0:Do_not_check 0x1: Fail_if_not_free - If the status of the MKey is not free, the UMR operation will fail. 0x2: Fail_if_free - If the status of the MKey is free, the UMR operation will fail.
	28	translation_offset_en	when set, translation_offset is valid (and not bsf_octword_acual_size)

Table 53 - UMR Control Segment Fields

Offset	Bits	Name	Description
4h	31:16	translation_octword_actual_size	Place for klm entries in 16B units (inline or pointers) KLM list should be aligned to 64B. SW can add PAD to the list of KLMs for this.
	15:0	bsf_octword_actual_size_or_translation_offset_15_0	Place for BSF entries in 16B units (inline or pointers). BSF list should be aligned to 64B. SW can add PAD to the list of BSFs for this. 16 LSB bits of translation_offset in 16B units is used to write klms/mtts at some offset from the start of the klm/mtt list describing the memory region. This enables changing only some of the klms/mtts of a region. translation_offset and size should be aligned to 64B

Table 53 - UMR Control Segment Fields

Offset	Bits	Name	Description
08h-0Ch	64	mkey_ctx_mask	mask bit for MkeyCtx bits. One bit per field Bit 0: length[63:0], len64 Bit 1: log_entity_size Bit 2: translations_octword_size Bit 3: mtt_first_index Bit 4: bsf_first_index Bit 5: bsf_octword_size Bit 6: start_addr Bit 7: pd (can only be used for UMRs posted to QPs with the REG_UMR service type) Bit 8: en_rinval Bit 9: expected_sigerr_count Bit 10: trast_a_en Bit 11: trast_b_en Bit 12: bsf enable Bit 13: mem_key[7:0] Bit 14: qpn Bit 15: version Bit 16: access_mode Bit 17: lr Bit 18: lw Bit 19: rr Bit 20: rw Bit 21: a Bit 22: umr_en Bit 23: small_fence_on_rdma_read_response Bit 24: goto_pci Bit 25: relaxed_ordering_write - supported only if HCA - CAP.relaxed_ordering_write_umr==1 Bit 26: no_snoop Bit 27: tlp_processing_hints[1:0] Bit 28: tlp_processing_hints_en Bit 29: free Bit 30: valid Bit 31: generation_counter[15:0] Bit 34: pasid Bit 41: tunneled_atomic Bit 47: relaxed_ordering_read - supported only if HCA - CAP.relaxed_ordering_read_umr==1
10h	26:0	translation_offset_42_16	MSB bits of translation offset. The MSB bits supported only when HCA_CAP.umr_extended_translation_offset==1. See Table 1571, “HCA Capabilities Layout,” on page 1671.

Table 54 - UMR Pointer Description Argument Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									0h	
mkey																															4h
address																															08h-0Ch

Table 55 - UMR Pointer Description Argument Fields

Offset	Bits	Name	Description
04h	31:0	mkey	
08h-0Ch	64	address	

Note that the address of the UMR pointer must be aligned to 2KB.

Memory buffer description is shown in Table 56.

Table 56 - UMR Memory Buffer (KLM) Description Argument Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											0h
byte_count																															04h	
mkey																															08h	
address																															08h-0Ch	

Table 57 - UMR Memory Buffer (KLM) Fields Description

Offset	Bits	Name	Description
00h	31:0	byte_count	Byte Count. Must be up to 2GB. Reserved when access mode==Fixed_Buffer_Size, and byte count is determined by log_entity_size in MKEY Context.

Table 57 - UMR Memory Buffer (KLM) Fields Description

Offset	Bits	Name	Description
04h	31:0	mkey	
08h-0Ch	64	address	

8.9.4.1.7 SET_PSV Segment

[TODO - move this table to UMR section, add table description the fields.](#)

Persistent Signature Variable (PSV) is configured by posting SET_PSV WR on SQ. Format of SET_PSV segment is shown in Table 58.

Table 58 - SET_PSV Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
								psv_index_or_tir_tis																								
psv_fields_or_tls_progress_params_nvmeotcp_progress_params																												04h-0Ch				

Table 59 - SET_PSV Segment Fields

Offset	Bits	Name	Description
0h	23:0	psv_index_or_tir_tis	If PSV, indicates psv_index If tls_progress_params, indicates TIR/TIS PSV index to be configured.
04h-0Ch	96	psv - fields_or_tls_pro- gress_params_nvme otcp_pro- gress_params	In case of PSV, then indicates PSV fields in offsets 0x4h-0xCh. See Table 690, “Persistent Signature Variable (PSV) Layout,” on page 899. In case of TLS_PROGRESS_PARAMS, then indicates TLS Progress Params fields. See Table 262, “TLS Progress Params Context Layout,” on page 528. In case of NVMEoTCP_PROGRESS_PARAMS, then indicates NVMEoTCP Progress Params fields. See Table 262, “TLS Progress Params Context Layout,” on page 528.

set_psv Segment specifies PSVs being configured. See Table 690, “Persistent Signature Variable (PSV) Layout,” on page 899. Please note that the pd field is reserved.

Execution of this command is subject to a check of the Protection Domain of the PSV and the SQ. If it does not match, its execution will be completed with error.

Table 61 - CHECK_PSV/RCHECK_PSV Segment Fields

Offset	Bits	Name	Description
14h	31: 0	L_Key	Local memory key where data is to be scattered
18h	31: 0	VA_h	Bits [63:32] of address within L_Key to scatter data
1Ch	31: 6	VA_l	Bits[31:6] or address within L_Key to scatter data
20h	23: 0	PSV0 index	index of PSV0
24h	23: 0	PSV1 index	index of PSV1
28h	23: 0	PSV2 index	index of PSV2
2Ch	23: 0	PSV3 index	index of PSV3

The format of CHECK_PSV response is shown in Table 703, “CHECK_PSV Response Layout”.

8.9.4.1.9 GET_PSV/RGET_PSV Segment

Table 62 - GET_PSV/RGET_PSV Segment Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h
																															04h
																															08h
																															0Ch
																															10h
																															14h
																															18h
																															1Ch
																															20h
																															24h
																															28h
																															2Ch

Table 63 - GET_PSV/RGET_PSV Segment Fields

Offset	Bits	Name	Description
10h	7:4	num_psv	Number of PSVs to be checked. Should be 1 - 4.
14h	31:0	L_Key	Local memory key where data is to be scattered
18h	31:0	VA_h	Bits [63:32] of address within L_Key to scatter data
1Ch	31:6	VA_l	Bits[31:6] or address within L_Key to scatter data
20h	23:0	PSV0 index	Index of PSV0
24h	23:0	PSV1 index	Index of PSV1
28h	23:0	PSV2 index	Index of PSV2

Table 63 - GET_PSV/RGET_PSV Segment Fields

Offset	Bits	Name	Description
2Ch	23:0	PSV3 index	Index of PSV3

The *data* part of the segment indicates the location in memory to place the output data. *psv_list* part of the segment contains the list of PSVs to be retrieved by the command.

The format of GET_PSV response is shown in Table 705, “GET_PSV Work Request Output”.

8.9.4.1.10 DUMP Work Request

Dump WQE should have at least one gather entry. Message size should be larger than 0.

Table 64 - DUMP Work Request Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	seg
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											
OPCODE = DUMP																															00	ctrl
																															0C	
data segment. See Section 8.9.4.1.15, “Send Data Segments,” on page 339																																data

8.9.4.1.11 Remote Address Segment

Table 65 - Remote Address Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00-04h
remote_virtual_address																															08	
rkey																															0C	

Table 66 - Remote Address Segment Field Descriptions

Offset	Bits	Name	Description
00h-04h	64	remote_virtual_address	

Table 66 - Remote Address Segment Field Descriptions

Offset	Bits	Name	Description
08h	31:0	rkey	Remote memory key

This segment will not be presented in DC and AV WQEs that will not result in packets sending (UMR, [GET_PSV](#), SET_PSV, NOP).

Note that atomic operation can be performed only on physical memory that is aligned to the atomic operation size, otherwise it will result with an error.

8.9.4.1.12 Atomic/Extended Atomic WQE and Segment

Atomic/extended atomic WQE will always contain atomic/extended atomic segment. It will also contain a single data (pointer) segment of the atomic operand size.

The 64-bit-argument atomic operations Fetch&Add and Compare&Swap use the Atomic Segment as described in Table 67. Fetch&Add uses only the first 64 bits for the add data. Compare&Swap uses the two 64 bits fields for the swap and compare data respectively.

The 64-bit-argument extended atomic operation Atomic Masked Fetch&Add uses the Atomic Segment (Table 67), where the first 64 bits provide the data to add, and the second 64 bits provide the field boundary.

Table 67 - Atomic Segment Format for 64-bit Atomic Argument

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00-04h
swap_or_add_data																															08-0Ch	
compare_data																																

Table 68 - Atomic Segment Format for 64-bit Atomic Argument Field Descriptions

Offset	Bits	Name	Description
0h-04h	64	swap_or_add_data	
08h-0Ch	64	compare_data	

The extended atomic operation Atomic Masked Compare&Swap uses the Extended Atomic Segment (Table 69) as described below:

Table 69 - Extended Atomic Segment (Masked Compare&Swap) Format for N Bytes Argument ($N \geq 8$)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												00h-(N-4)h
swap_data																																	
compare_data																												N-(2N-4)h					
swap_mask																												2N-(3N-4)h					
cmp_mask																												3Nh-(4N-4)h					

Table 70 - Extended Atomic Segment (Masked Compare&Swap) for N Bytes Argument ($N \geq 8$) Field Descriptions

Offset	Bits	Name	Description
00h-(N-4)h	8*N	swap_data	
N-(2N-4)h	8*N	compare_data	
2N-(3N-4)h	8*N	swap_mask	

Table 70 - Extended Atomic Segment (Masked Compare&Swap) for N Bytes Argument ($N \geq 8$) Field Descriptions

Offset	Bits	Name	Description
3Nh- (4N-4)h	8*N	compare_mask	

The extended atomic operation Atomic MultiField Fetch&Add uses the Extended Atomic Segment (Table 69) as described below:

Table 71 - Extended Atomic Segment (MultiField Fetch&Add) Format for N bytes Argument ($N \geq 8$)

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-(N-4)h
add_data																																
field_boundary																												Nh-(2N-4)h				

Table 72 - Extended Atomic Segment (MultiField Fetch&Add) for N Bytes Argument ($N \geq 8$) Field Descriptions

Offset	Bits	Name	Description
00h- (N-4)h	8*N	add_data	
Nh- (2N-4)h	8*N	field_boundary	

The following are the segments for 4-byte atomics.

Table 73 - Extended Atomic Segment (Masked Compare&Swap) Format for 32-bit Argument

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											04
swap_data																														0C		
compare_data																														14		
swap_mask																														1C		
cmp_mask																																

Table 74 - Extended Atomic Segment (Masked&Swap) for 32-bit Argument Field Descriptions

Offset	Bits	Name	Description
04h	31:0	swap_data	
0Ch	31:0	compare_data	
14h	31:0	swap_mask	
1Ch	31:0	cmp_mask	

Table 75 - Extended Atomic Segment (MultiField Fetch&Add) Format for 32-bit Argument

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00
add_data																												04			
field_boundary																												08-0C			

Table 76 - Extended Atomic Segment (MultiField Fetch&Add) for 32-bit Argument Field Descriptions

Offset	Bits	Name	Description
00h	31:0	add_data	
04h	31:0	field_boundary	

Atomic operations of 1, 2 bytes are implemented using a 32-bit argument, using mask in the COMPARE&SWAP fields of the atomic message. For more details, see Section 24, “Extended Atomic Operations,” on page 1039.

The Atomic operand length is set by the OPC_MOD field in the WQE control segment. If the OPC_MOD indicates indirect (pointer) reference to atomic argument, the Atomic Segment specifies the single pointer that refers to atomic segment in memory. The pointer follows the data segment format shown on Table 80, “Memory Pointer Data Segment Format” on page 340. Length of the atomic argument is specified implicitly by *byte_count* field of indirect atomic segment.

8.9.4.1.13 Vector Calc Segment

Table 77 - Vector Calc Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h	
calc_operation_0								calc_operation_1								calc_operation_2								calc_operation_3								
																																04h-08h
tag_size	tag_type		operand_size		operand_type		calc_matrix_exist	little_endian	tag_exist	calc_matrix_type		chunk_size									num_of_vectors								0Ch			
																																10h
calc_matrix_key																																14h
calc_matrix_address																																18h-1Ch
	vector_size																															20h

Table 77 - Vector Calc Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									24h
vectors_key																														28h-2Ch
vectors_address																														

Table 78 - Vector Calc Segment Fields

Offset	Bits	Name	Description
00h	31:24	calc_operation_0	calc operation 0. The supported operations reported in Vector calc capabilities by calc_operation_0. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772 0x0: NO_CALC_OP 0x1: ADD_OP 0x2: MAX_OP 0x3: AND_OP 0x4: OR_OP 0x5: XOR_OP 0x6: MIN_OP 0x7: SWAP_ENDIANNESNESS_OP
	23:16	calc_operation_1	calc operation 1. The supported operations reported in Vector calc capabilities by calc_operation_1. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772 0x0: NO_CALC_OP 0x5: XOR_OP
	15:8	calc_operation_2	calc operation 2. The supported operations reported in Vector calc capabilities by calc_operation_2. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772 0x0: NO_CALC_OP 0x5: XOR_OP
	7:0	calc_operation_3	calc operation 3. The supported operations reported in Vector calc capabilities by calc_operation_3. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772 0x0: NO_CALC_OP 0x5: XOR_OP

Table 78 - Vector Calc Segment Fields

Offset	Bits	Name	Description
0Ch	31:30	tag_size	Size of tag that may be associated with each data element. All elements either have or do not have such a tag depending on tag_exist field. 0x0: tag_4_bytes 0x1: tag_8_bytes
	29:28	tag_type	Tag data type 0x0: tag_unsigned_integer 0x1: tag_signed_integer 0x2: tag_float
	27:26	operand_size	Size of each operand 0x0: op_4_bytes 0x1: op_8_bytes
	25:24	operand_type	Data type 0x0: op_unsigned_integer 0x1: op_signed_integer 0x2: op_float
	23	calc_matrix_exist	If set, calc matrix exist. Supported when calc_matrix==1 in Vector calc capabilities. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772
	22	little_endian	Assume data is little endian. Swap bytes before calc operation, and swap bytes back after the operation.
	21	tag_exist	A tag exists with the data.
	20	calc_matrix_type	Galois multipliers type: 0x0: 4bits 0x1: 8bits - Supported only when calc_matrix_type_8bits==1 in Vector calc capabilities. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772.
	18:16	chunk_size	Output data will be interleaved in chunks of 64*(2^chunk size). The maximum supported size is reported by max_chunk_size in Vector calc capabilities. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772
	7:0	num_of_vectors	Number of vectors. The maximum supported number of vectors is reported by max_vector_count in Vector calc capabilities. See Section 29.3.4.8, “Vector Calc Capabilities,” on page 1772
14h	31:0	calc_matrix_key	Local key address of the calc matrix. valid only if calc_matrix_exist==1.
18h-1Ch	64	calc_matrix_address	Virtual address of the calc matrix. valid only if calc_matrix_exist==1.
20h	30:0	vector_size	Vector size in bytes. Must be in granularity of 64*(2^chunk size).

Table 78 - Vector Calc Segment Fields

Offset	Bits	Name	Description
24h	31:0	vectors_key	Local key address where the calc vectors allocated.
28h-2Ch	64	vectors_address	Virtual address where the calc vectors allocated.

8.9.4.1.14 Dual Write WQE

Dual Write Descriptor will specify two addresses to be used for remote write operation. Dual-write descriptor structure is shown in Table 79. **OPCODE** field will carry *Write* opcode. Grey area indicates fields from standard WQE format.

Table 79 - Dual-write Descriptor Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
CTRL segment																															00h-0Ch	
Second_Remote Virtual Address (high) [63:32]																															10h	
Second_Remote Virtual Address (low) [31:0]																															14h	
Second_Remote Memory Handle (Rkey)																															18h	
																															1Ch	
Remote Address																															20-2C	
Gather list																															-	

8.9.4.1.15 Send Data Segments

Send Data segments compose gather lists. There are two types of data segments: memory pointers and inline data. They are distinguished by the value of inline field (bit 31 at offset 0), as shown in the following tables. The gather list length is calculated according to **DS** (in the *Ctrl* segment) of the descriptor.

The format of the *data* segment that uses memory pointers for data is shown in Table 80, “Memory Pointer Data Segment Format”.

Table 80 - Memory Pointer Data Segment Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2											
is inline	byte_count																												00	
l_key																												04		
local_address																												08-0Ch		

Table 81 - Memory Pointer Data Segment Field Descriptions

Offset	Bits	Name	Description
0h	31	is_inline	Must be 0 for Memory pointer data segment.
	30:0	byte_count	
04h	31:0	l_key	
08h-0Ch	64	local_address	

Memory pointer segments always use 16 bytes. Multiple memory pointer segments can be present in the WQE. However, UMR or atomic WQEs only support a single memory pointer in their gather list. A byte count value of zero means 2Gbyte data transfer. To send a request with zero data(send/RDMA-write with zero length or RDMA-read request with zero length), **DS** should be set such that no data segments appear in the WQE.

Table 82, “Inline Data Segment Format” shows the format of *data* segments containing data.

Table 82 - Inline Data Segment Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2											
is inline																					byte_count							00		
data_and_padding[0]																											04h			
data_and_padding[1]																											08h			
...																											0Ch-...			

Table 83 - Inline Data Segment Field Descriptions

Offset	Bits	Name	Description
0h	31	is_inline	Must be 1 for Inline data segment.
	9:0	byte_count	Byte count of data in granularity of byte.
04h- ...	31:0	data_and_padding[...]	Data byte of size reported by byte_count. The last octoword should be padded if the byte_count does not cover all of its 16 bytes.

Every data segment containing data occupies one or more octowords (16-byte chunks). The amount of data to be transferred is specified by the **byte_count** field, and the length of a data segment depends on the value of this field. The last octoword should be padded if the **byte_count** does not cover all of its 16 bytes. **byte_count** should not exceed the WQE length beyond that specified in the **DS** field of the WQE. If **byte_count** field is zero, no data will be added to the message. SW which plan to work with a simple model where all WQEs are with fixed size, can set inline segments with zero byte count to extend send WQE.

Data is filled in using Big Endian order (the first byte of data occupies bits 31:24 of the field, and padding - if needed - is added in the least significant bytes).

8.9.4.1.16 Receive Data Segments

Receive Data segments compose scatter lists that can be only of memory pointer type. The format of the *receive data* segment is similar to the Send Data Segment in Memory Type except for the **start_padding** field as shown in Table 84, "Receive Data Segment Format".

Table 84 - Receive Data Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											00h
start padding	byte_count																															
	l_key																														04h	
	local_address																														08h-0Ch	

Table 85 - Receive Data Segment Field Descriptions

Offset	Bits	Name	Description
0h	31	start_padding	
	30:0	byte_count	

Table 85 - Receive Data Segment Field Descriptions

Offset	Bits	Name	Description
04h	31:0	l_key	
08h-0Ch	64	local_address	

Receive Data Segment always use 16 bytes. Multiple memory pointer segments can be present in the WQE. A byte count value of zero means 2Gbyte.

start_padding can be set to enable Start Padding as explained in Section 4.3.14, “Start of Packet Padding (RX),” on page 205.

8.9.4.1.17 Tag Matching Segment

This segment is used to perform a list of operations on XRQ matching list. The segment should be added after the WQE control segment. Tag matching segment size is 2DS, regardless of the list of operations.

Table 86 - Tag Matching Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset																										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h																									
opcode								xtq_cqe_req	sw_cnt_enable							index																																									
																sw_cnt																																									
append_tag																																																									
append_mask																																																									

Table 87 - Tag Matching Segment Fields

Offset	Bits	Name	Description
00h	31:28	opcode	Tag matching operation opcode 0x0: NOP 0x1: Append 0x2: Remove
	23	xrq_cqe_req	Request responder CQE on operation. CQE will be generated on the corresponding XRQ CQ.
	22	sw_cnt_enable	If set, XRQ's <i>sw_phase_cnt</i> will be set to <i>sw_cnt</i> value. <i>sw_cnt</i> will be set in the end of the list operation.
	15:0	index	For append opcode - next index to be reserved for next appended tag. First append will be to index 0. For remove opcode - index to be removed.
04h	15:0	sw_cnt	<i>sw_cnt</i> to be set for XRQ's <i>sw_phase_cnt</i> .
10h-14h	64	append_tag	Valid only in Append operations.
18h-1Ch	64	append_mask	Bitmask for tag matching operation. If bit is set to 0 then corresponding bit in <i>append_tag</i> will not take part in the matching logic. Valid only in Append operations.

8.9.4.1.18 MLX WQE Format

The format of the MLX segment is shown below in Table 88. MLX WQE consists of control segment, MLX segment and data segment. The WQE data segment represent a full IB packet, starting from LRH and ending with 4 bytes of ICRC, or a placeholder for ICRC. VCRC is always appended to the packet by the hardware. The hardware overwrites the last 4 bytes with a valid ICRC if the *icrc* bit in the ctrl segment is cleared.

When MLX WQE is used to generate Raw IB packets, the ICRC is not part of the packet. Therefore, the last 4 bytes of the payload are the actual last four bytes of the payload (and not the placeholder for the ICRC) and the *icrc* bit must be set.

The MLX segment is shown below in Table 88.

Table 88 - MLX Segment Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 88 - MLX Segment Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																dlid																0ch

The following table explains the MLX WQE specific fields description. General fields are explained in Table 43, “General Ctrl Segment Field,” on page 313.

Table 89 - MLX Segment Format Field Descriptions

Offset	Bits	Name	Description
00h	31:0	reserved	

Table 89 - MLX Segment Format Field Descriptions (Continued)

Offset	Bits	Name	Description
04h	30	vl15	Specifies whether this message should be sent on virtual lane 15 to the fabric. 1 - send packet on VL15 (packet will not undergo SL2VL mapping). The packet VL is 15 0 - send packet on another VL (packet will undergo SL2VL mapping). The packet VL is one of 0-14
	29	slr	Specifies whether packet SLID MSBits need to be taken from the portInfo or from the packet 0 - SLID MSBits are taken from portInfo 1 - SLID MSBits are taken from the packet (as appears in the payload of the MLX WQE or pointed to by the MLX WQE gather list) In any case, the lowermost LMC bits of the SLID (of the generated packet) are taken from the payload. Setting the SLR to 1 should be used when transmitting SMPs with the permissive SLID.
	27	icrc	0 - Don't add ICRC field (raw packet, or ICRC calculated by driver) 1 - Add CRC DW to the packet
	26	fl	Force Loopback. If set, the packet will be sent to loopback interface.
	23:20	static_rate	Maximum injection rate to be used while scheduling this message for transmission: 7: 2.5 Gb/s 8: 10 Gb/s 9: 30 Gb/s 10: 5 Gb/s 11: 20 Gb/s 12: 40 Gb/s 13: 60 Gb/s 14: 80 Gb/s 15: 120 Gb/s <u>17: Gbps_56</u> <u>20: Gbps_25</u> <u>21: Gbps_100</u> <u>22: Gbps_200</u> <u>25: Gbps_50</u> ~~other reserved~~ <TBD> see if need extensions for EDR
	19:16	sl	Specifies the service level modifier to be used for SL to VL mapping in the IB port
08h	31:0	reserved	
0ch	15:0	dlid	Destination LID for this packet. The same value should also be written to memory in the LRH.Dlid field of the packet

Data segments are explained in “Send Data Segments” on page 339. Data segments should include 4 bytes at their tail for ICRC (non Raw packets only). The extra 4 bytes should be accounted for in the DS field of the WQE.

8.9.4.1.19 MMO Metadata Segment

This segment is used in Memory to Memory Offloads (MMO) as described in TBD.

Table 90 - MMO MetaData Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
mmo_control_31_0																															04h	
local_key																															08h-0Ch	
local_address																																

Table 91 - MMO MetaData Segment Fields

Offset	Bits	Name	Description
00h	31:0	mmo_control_31_0	LSB of mmo_control. The MSB is located in ctrl_general_id in Ctrl Segment
04h	31:0	local_key	Local Mkey. The size of the metadata is defined according to the MMO type.
08h-0Ch	64	local_address	Local Virtual Address

8.9.4.1.20 ASO CTRL Segment

The ASO Access WQE allows to accesses to ASO Context: Read and Conditional Update (both can be performed by a single WQE).

For Read, the ASO Ctrl segment includes the memory location where the ASO context will be written.

For Conditional Update the ASO Ctrl segment includes:

- The data modified and its location in the ASO context in case the update condition succeeds

- The update condition
 - 2 sub-conditions
 - Each sub-condition can compare different parts of the ASO context to different values
 - The logic between the sub-conditions

Table 92 - ASO CTRL Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset								
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	va_h																00h
va_l																												read_enable	04h									
l_key																														08h								
data_mask_mode											condition_0_operand	condition_1_operand	condition_0_offset	condition_1_offset	condition_operand															data_offset	0Ch							
condition_0_data																														10h								
condition_0_mask																														14h								
condition_1_data																														18h								
condition_1_mask																														1Ch								
bitwise_data																														20h-24h								
data_mask																														28h-2Ch								

Table 93 - ASO CTRL Segment Fields

Offset	Bits	Name	Description
00h	31:0	va_h	Bits [63:32] of address within l_key to scatter the ASO Context. Valid only when read_enable is set.
04h	31:6	va_l	Bits [31:6] of address within l_key to scatter the ASO Context. Bits[5:0] are assumed to be 0x0. Valid only when read_enable is set.
	0	read_enable	If set, ASO Context will be scattered to the memory location indicated by l_key and {va_h, va_l, 6'b0}
08h	31:0	l_key	Local Memory Key where the ASO Context will be scattered. Valid only when read_enable is set
0Ch	31:30	data_mask_mode	Indicates the width of the ASO modification, and the behavior of the modification mask 0x0: BITWISE_64BIT - update up to 64 bit. Bitwise mask allows to choose which bit to update 0x1: BYTEWISE_64BYTE - update upto 64 Byte. Bytewise mask allows to choose which Byte to update 0x2: CALCULATED_64BYTE - same as BYTEWISE_64-BYTE, however the mask will be logically Ored with a HW calculated mask
	23:20	condition_0_operand	Defines the operation between the masked (condition_0_mask) ASO context data (condition_0_offset) and the masked (condition_0_mask) condition 0 data (condition_0_data) 0x0: ALWAYS_FALSE 0x1: ALWAYS_TRUE 0x2: EQUAL - == 0x3: NOT_EQUAL - != 0x4: GREATER_OR_EQUAL - >= 0x5: LESSER_OR_EQUAL - <= 0x6: LESSER - < 0x7: GREATER - > 0x8: CYCLIC_GREATER 0x9: CYCLIC_LESSER other values are reserved

Table 93 - ASO CTRL Segment Fields

Offset	Bits	Name	Description
0Ch	19:16	condition_1_operand	Defines the operation between the masked (condition_1_mask) ASO context data (condition_1_offset) and the masked (condition_1_mask) condition 1 data (condition_1_data) 0x0: ALWAYS_FALSE 0x1: ALWAYS_TRUE 0x2: EQUAL - == 0x3: NOT_EQUAL - != 0x4: GREATER_OR_EQUAL - >= 0x5: LESSER_OR_EQUAL - <= 0x6: LESSER - < 0x7: GREATER - > 0x8: CYCLIC_GREATER 0x9: CYCLIC_LESSER other values are reserved
	15:12	condition_0_offset	Defines the offset inside ASO Context of the data that will be used as the term left of the operand (condition_0_operand) when calculating Condition 0. Value is given in units of 4B (32bit). e.g. IPSEC ASO valid bit is in bit[31] of offset ~~45~~ <u>(0xF)0x0</u>
	11:8	condition_1_offset	Defines the offset inside ASO Context of the data that will be used as the term left of the operand (condition_1_operand) when calculating Condition 1. Value is given in units of 4B (32bit). e.g. IPSEC ASO valid bit is in bit[31] of offset ~~45~~ <u>(0xF)0x0</u>
	7:6	condition_operand	Defines the operand between Condition 0 and Condition 1. when the result is TRUE the update will be performed. otherwise the ASO context will remain unchanged 0x0: LOGICAL_AND - & 0x1: LOGICAL_OR - other values are reserved
	2:0	data_offset	Defines the offset inside the ASO context of the data that will be modified, if the update condition succeeds. Value is given in units of 8B (64bit)
10h	31:0	condition_0_data	The data that will be used as the term right of the operand (condition_0_operand) when calculating Condition 0
14h	31:0	condition_0_mask	Defines the mask applied to condition_0_data and ASO context data before calculating Condition 0
18h	31:0	condition_1_data	The data that will be used as the term right of the operand (condition_1_operand) when calculating Condition 1
1Ch	31:0	condition_1_mask	Defines the mask applied to condition_1_data and ASO context data before calculating Condition 1

Table 93 - ASO CTRL Segment Fields

Offset	Bits	Name	Description
20h-24h	64	bitwise_data	New ASO context data (at offset data_offset) if the update condition succeeds. Valid only if data_mask_mode is BITWISE_64BIT. For other modes, the data will reside in the ASO Data segment
28h-2Ch	64	data_mask	Mask defining which bits of the ASO context (at offset data_offset) will be updated if the update condition succeeds. For data_mask_mode BITWISE_64BIT, each bit in the mask represents a bit in the data. Set bit indicates it will be modified. For data_mask_mode BYTEWISE_64BYTEFor data_mask_mode BYTEWISE_64BYTE and CALCULATED_64BYTE, each bit in the mask represents a Byte in the data. Set bit indicates the Byte will be modified.

8.9.4.1.21ASO Data Segment

The ASO Data Segment is required when it is desired to modify more than 64bit of the ASO context using a single WQE. This is indicated by using data_mask_mode BYTEWISE_64BYTE in the ASO Ctrl segment. Otherwise this segment is not required.

Table 94 - ASO Data Segment Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-3Ch
bitwise_data																																

Table 95 - ASO Data Segment Fields

Offset	Bits	Name	Description
00h-3Ch	512	bitwise_data	New ASO context data (at offset data_offset), if the update condition succeeds. Valid only if data_mask_mode is BYTEWISE_64BYTE, otherwise the entire segment should not be used.

8.9.4.1.22 Flow Update Control Segment

Table 96 - Flow Update Ctrl Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
			update_opcode					flow_index																				00h				
								destination_handle																				04h				
																															08h-2Ch	

Table 97 - Flow Update Ctrl Segment Field

Offset	Bits	Name	Description
00h	28	update_opcode	Indicates which operation to perform: 0x0: SET_FLOW - If a flow is found with identical match value, the flow will be updated, otherwise the Flow is added in an empty slot. 0x1: DISABLE - If a flow is found with identical match value, the flow will be invalidated, and the slot will be marked as empty.
	23:0	flow_index	The location of the Flow inside the Flow Group. valid only for Flow Groups with <i>update_flow_lookup_mod</i> INDEX_BASED.
04h	28:0	destination_handle	Indicates the destination for packets that match this Flow. The handle is built of 2 parts: destination_type and destination_id. the QUERY_HCA_CAP.flow_table_update_wqe_dest_type_offset indicates the number of bits dedicated for destination_id, and the bit offset at which the destination_type starts. Details can be found in Table 98, "Destination Handle". Note that the width of the <i>destination_type</i> must be not sufficient, otherwise this destination cannot be used for this method of Flow update. <u>Valid only for update_opcode SET_FLOW</u>

Table 98 - Destination Handle

Destination Type	destination_type value	destination_id description
VPort	0x0	holds the VPort number in bits[15:0] other bits are reserved. The Vport is local to this Eswitch Owner.

Table 98 - Destination Handle

Destination Type	destination_type value	destination_id description
Flow Table	0x1	holds the Flow Table ID
TIR	0x2	holds the TIR number
Drop MISC	0x7	0x0: DROP 0x1: ALLOW

8.9.4.1.23 Flow Update Data Segment

Table 99 - Flow Update Data Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								flow_counter_id																				00h				
																																04h-10h
action_list																																14h-1Ch
match_value																																20h-3C

Table 100 - Flow Update Data Segment Field

Offset	Bits	Name	Description
00h	23:0	flow_counter_id	Flow Counter handle. Value 0x0 indicates a NULL counter <u>Valid only for update_opcode SET_FLOW</u>
14h-1Ch	96	action_list	Holds a list, built out of the actions defined in Table 101, “Action List Entries”. These actions will be executed if a packet matches this Flow. <u>Valid only for update_opcode SET_FLOW</u>

Table 100 - Flow Update Data Segment Field

Offset	Bits	Name	Description
20h-3Ch	256	match_value	Values of the packet headers and parameters that should match the Flow. The format of this field is identical to the format of the Match Definer <i>flow_match_definer_id</i> of the Flow Group. Invalid fields (for which the Match Definer has mask bits set to 0x0) must be set to 0x0 in the match_value.

Table 101 - Action List Entries

Action Name	Action ID	Action Size in DW	Action Description	Action Layout
NOP	0x0	1	No action is performed	Table 102, “NOP Action Layout”
FLOW_TAG	0xC	1	Set the Flow Tag which will be reported in the CQE	Table 104, “FLOW_TAG Action Layout”

Table 102 - NOP Action Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
action_id																																00h

Table 103 - NOP Action Fields

Offset	Bits	Name	Description
00h	31:24	action_id	See Table 101, “Action List Entries”

Table 104 - FLOW_TAG Action Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
action_id								flow_tag																							00h	

Table 105 - FLOW_TAG Action Fields

Offset	Bits	Name	Description
00h	31:24	action_id	See Table 101, “Action List Entries”
	23:0	flow_tag	Flow Tag new value

8.9.4.2 Send WQE Construction Summary

Table 106, “WQE Construction Summary,” on page 355 summarizes how to construct WQEs for the various transport services and opcodes.

Notes:

- Segments marked with “V” are mandatory.
- Segments marked with “Ptr” are data segments which support pointer only. Data segments that are marked with “V” support both memory pointer and inline data segments.
- Segments marked with “-” are not part of the WQEs.
- Fields marked with “XoIB” are placed for QPs that offload IPoIB or EoIB otherwise are not part of the WQE
- The order of the segments to place in the WQEs are from left to right column.

The table refers to the following WQE segment types:

- CTRL - Section 8.9.4.1.1, “Ctrl Segment,” on page 312
- AV - Section 8.9.4.1.2, “Datagram Segments,” on page 317 and Section 8.9.4.1.3, “DC Transport Segment,” on page 317
- XRC - Section 8.9.4.1.4, “XRC Segment,” on page 317
- Eth - Section 8.9.4.1.5, “Eth Segment and Padding Segment,” on page 318
- **MLX** - Section 8.9.4.1.18, “MLX WQE Format,” on page 343
- PAD + InnerEth - Section 8.9.4.1.5, “Eth Segment and Padding Segment,” on page 318
- RDMA - Section 8.9.4.1.11, “Remote Address Segment,” on page 331
- Wait - [<TBD>](#)
- PSV - Section 8.9.4.1.7, “SET_PSV Segment,” on page 327, [Section 8.9.4.1.8, “CHECK_PSV/rCHECK_PSV Segment,” on page 328](#), [Section 8.9.4.1.9, “GET_PSV/RGET_PSV Segment,” on page 330](#).
- Atomic - Section 8.9.4.1.12, “Atomic/Extended Atomic WQE and Segment,” on page 332
- UMR - Section 8.9.4.1.6, “User-mode Memory Registration (UMR) WQE Format,” on page 322
- Data - Section 8.9.4.1.15, “Send Data Segments,” on page 339

- ASO Ctrl
- ASO data
- Flow Update Ctrl
- Flow Update Data

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
Nop	RC UC UD DC XRC MLX Eth IpoIB	V	-	-	-	-		-
SendInv	RC	V	-	-	-	-		V
	XRC	V	XRC	-	-	-		V
	DC	V	AV	-	-	-		V
RDMA Write/ WriteImm	RC UC	V	-	-	RDMA	-		V
	XRC	V	XRC	-	RDMA	-		V
	DC	V	AV	-	RDMA	-		V
Send/ Dump	RC UC	V	-	-	-	-		V
	XRC	V	XRC	-	-	-		V
	UD	V	AV	XoIB	-	-		V
	DC	V	AV	-	-	-		V
	MLX	V	MLX	-	-	-		V
	IPoIB	V	AV	V	-	-		V

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
Send/ ENHAN CED_M PSW	Eth	V	Eth	-	-	-		V
Dump	Eth	V	-	-	-	-		V
SendImm	RC UC	V	-	-	-	-		V
	XRC	V	XRC	-	-	-		V
	UD DC	V	AV	-	-	-		V
SendLSO	UD	V	AV	XoIB must	-	-		V
	IPoIB	V	AV	V	-	-		V
	Eth	V	Eth	-	-	-		V
Wait	RC UC	V	-	-	-	Wait		-
	UD	V	AV	XoIB	-	Wait		-
	DC	V	AV	-	-	Wait		-
	XRC	V	XRC	-	-	Wait		-
	MLX(TBD)	V	MLX	-	-	Wait		-
	Eth(TBD)	V	Eth	-	-	Wait		-
	IPoIB	V	AV	V	-	-Wait		-
RDMA Read	RC	V	-	-	RDMA	-		Ptr
	DC	V	AV	-	RDMA	-		Ptr
	XRC	V	XRC	-	RDMA	-		Ptr

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
Atomic C&S, F&A, Msk C&S Msk F&A	RC	V	-	-	RDMA	Atomic		Ptr
	DC	V	AV	-	RDMA	Atomic		Ptr
	XRC	V	XRC	-	RDMA	Atomic		Ptr
Receive/ Send Enable	RC UC	V	-	-	-	Wait		-
	UD	V	AV	XoIB	-	Wait		-
	DC	V	AV	-	-	Wait		-
	XRC	V	XRC	-	-	Wait		-
	MLX	V	MLX	-	-	Wait		-
	Eth	V	Eth	-	-	Wait		-
	IPoIB	V	AV	V	-	-Wait		-

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
Set PSV/ GetPSV/ RGetPSV/ ChkPSV/ RChkPSV	RC UC	V	-	-	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-
	UD	V	AV	XoIB	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-
	DC	V	AV	-	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-
	XRC	V	XRC	-	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-
	MLX	V	MLX	-	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-
	Eth	V	-	-	-	Set PSV/ GetPSV/ GetPSV/ ChkPSV/ ChkPSV		-

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
UMR	RC UC UD DC XRC MLX Eth IPoIB	V	-	-	-	UMR		-
Tag Matching	RC	V				Tag Match- ing		Only for Appen d oper- ations
LOAD_R EMOTE_ MICRO_ APP	RC	V	-	-	-	-		Data+P tr
	DC	V	AV	-	-	-		Data+P tr
	XRC	V	XRC	-	-	-		Data+P tr
STORE_ REMOTE_ MI- CRO_AP P	RC UC	V	-	-	-	-		Data+P tr
	DC	V	AV	-	-	-		Data+P tr
	XRC	V	XRC	-	-	-		Data+P tr
MMO	Eth	V					V	2 Data Seg- ments
ACCESS _ASO	Eth	V	ASO CTRL	-	-	ASO DATA	-	-

Table 106 - WQE Construction Summary

OpCode	Transport	CTRL	AV/ XRC/ Eth/ ASO CTRL/ Flow Update Ctrl/ MLX	PAD+ Inner Eth	RDMA	Wait/ PSV/ Atomic/ UMR/ Tag Matching /ASO Data/ Flow Update Data	MMO MetaDa ta Segmen t	Data
Flow Table Update	Eth	V	Flow Update Ctrl	-	-	Flow Update Data	-	-
<u>CALC -</u> <u><TBD></u>	<u>[TBD] CALC is subset of SEND, SENDwINV, SENDwIMM, WRITE, WRITEwIMM,</u> <u>based on opcode modifier. See also DOC71679. TBD up to 16 args</u>							

TBD: Need to see if we want to add special QP types for UD EoIB/IPoIB. Currently they are part of UD QPs with “XoIB” note.

8.9.4.3 Receive WQE Format

Receive WQE is built of 16-byte chunks, each one containing scatter pointer as shown in Table 84, "Receive Data Segment Format". The number of scatter pointers can vary in different WQEs, and it is controlled as explained in “Receive Queue” on page 302.

If RQ is subject for WQE signature, first segment is a WQE signature segment (Table 38, "Receive Queue - Signature Entry Format").

8.9.4.4 Shared Receive WQE Format

Next segment of Shared Receive WQE format is shown in Table 107.

Table 107 - Shared Receive WQE Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																next_wqe_index												00h				
signature																											04h					
																																08h-0Ch
receive_data_segment[0] (See Table 84, “Receive Data Segment Format,” on page 341)																																10h-1Ch
receive_data_segment[1] (See Table 84, “Receive Data Segment Format,” on page 341)																																20h-2Ch
...																																30h-:

The WQE on SRQ is built of octowords. The first one is the *next* segment, followed by a scatter list for an incoming message. The present of the next segment is decided according to HCA - CAP.basic_cyclic_rcv_wqe and RMP/XRC_SRQ.basic_cyclic_rcv_wqe. The format of the *receive data* segment is shown in Table 84. The number of scatter pointers can vary in different WQEs, and is controlled as explained in “Receive Queue” on page 302.

Table 108 - Shared Receive WQE Format Field Descriptions

Offset	Bits	Name	Description
00h	15: 0	next_wqe_index	Index (pointer) in the WQE buffer to the next WQE to be executed. The pointer is in SRQ stride units. Reserved for continuous SRQ
04h	31: 24	signature	WQE signature. The signature covers entire WQE as well as the <i>SRQ number and WQE index</i> corresponding to that WQE. Should result in 0xff. This feature is enabled per SRQ. The signature covers the whole WQE stride including the NULL segments See Section 17.2.3, “Work Queues Elements Signature,” on page 879
10h-...	128	receive_data_segment[...] (See Table 84, “Receive Data Segment Format,” on page 341)	Receive data segment/

8.9.5 UD Address Vector Format

The format of a UD Address Vector for InfiniBand link layer is shown in Table 109.

Table 109 - UD Address Vector Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dc_key[63:32] / q_key																															00h	
dc_key[31:0]																															04h	
ext				reverse_sl				destination_qp_or_dct																							08h	
static_rate			sl_or_eth_port				fl	mlid					rlid_or_udp_sport														0Ch					
reserved																															10h	
rmac[47:16]																															14h	
rmac[15:0]															tclass							hop_limit									18h	
grh				src_addr_index								flow_label																				1Ch
rgid_or_rip																															20h-2Ch	

Table 110 defines fields of the UD Address Vector.

Table 110 - UD Address Vector Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	dc_key[63:32] / q_key	For DC transport - DC Access key For UD transport - Q_key Otherwise - reserved	
04h	31:0	dc_key[31:0]	For DC transport - DC Access Key Otherwise - reserved	

Table 110 - UD Address Vector Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	ext	Address Vector extension exist. When HCA_CAP.compact_address_vector==0, then ext Must be set to 1. When IpoIB or EthoIB, ext Must be set to 1. When ext ==0, Address Vector is compact without the global extention. i.e. size of the UD address vector is only the first 16 bytes.	
	27:24	reverse_sl	Reverse SL for data responses. Supported only when HCA_CAP.dc_v2==1.	
	23:0	destination_qp_or_dc_t	For DC transport - DCT For UD transport - QP number	

Table 110 - UD Address Vector Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:28	static_rate	Maximum static rate control. 0: No limit on static rate (100% port speed) 1-6: reserved 7: 2.5 Gb/s 8: 10 Gb/s 9: 30 Gb/s 10: 5 Gb/s 11: 20 Gb/s 12: 40 Gb/s 13: 60 Gb/s 14: 80 Gb/s 15: 120 Gb/s <u>17: Gbps_56</u> <u>20: Gbps_25</u> <u>21: Gbps_100</u> <u>22: Gbps_200</u> <u>25: Gbps_50</u>	
	27:24	sl_or_eth_prio	SL/Ethernet Priority For InfiniBand This field encodes the Service Level (SL) For best performance, recommended to send UD messages on the SL written in the QP. for DC, this field must be equal to QP.SL. <u>[TBD] Check if this is recommendation is relevant also for RoCE</u> For RoCE: eth_prio [27:25]-ethernet priority, eth_prio[24]-dei/cfi When f_eth_prio is enabled priority bits are taken from QP's Address Path.	
	23	fl	Force Loopback. For IB Only.	
	22:16	mlid	For InfiniBand: Source LID - the lower LMC bits (upper bits are taken from portInfo) Otherwise reserved	
	15:0	rlid_or_udp_sport	For InfiniBand: Remote (Destination) LID For R-RoCE v2.0: UDP source port which must belong to the range: [QUERY_HCA_CAP.r_roce_udp_src_port_range_min-QUERY_HCA_CAP.r_roce_udp_src_port_range_max]. Valid only when HCA_CAP.RoCE_CAP.sw_r_roce_src_udp_port==1. For RoCE v1.0 and v1.5 this field is reserved.	
10h	31:0	reserved		
14h	31:0	rmac[47:16]	For RoCE: Remote (destination) MAC[47:16] Otherwise reserved	

Table 110 - UD Address Vector Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:16	rmac[15:0]	For RoCE: Remote (destination) MAC[15:0] Otherwise reserved	
	15:8	tclass	For InfiniBand and RoCE: GRH TClass <TBD> need to see re ECN bits For R-RoCE: IPv6 TClass or IPv4 ToS When f_dscp in QP's Address Path is enabled DSCP bits are ignored and taken from QP's Address Path. When f_ecn is enabled in QP's Address Path ECN bits are ignored and taken from QP's Address Path.	
	7:0	hop_limit	For InfiniBand and RoCE: GRH Hop Limit For RRoCE: IPv6 hop limit or IPv4 TTL	
1Ch	31	reserved		
	30	grh	For InfiniBand, if set, GRH will be placed in the packet header, if clear, no GRH. Reserved for RoCE For RoCE L3 type is set according to address table entry pointed by addr_index field.	
	27:20	src_addr_index	For InfiniBand Index to GID in the port to get source GID address. For RoCE index to RoCE Address table in the port to get the source GID/IP, MAC addresses, VID (if available), L3 version and RoCE protocol version.	
	19:0	flow_label	IPv6 flow label	
20h-2Ch	128	rgid_or_rip	Remote GID/IP address. For IB port - Valid if the GRH bit is set. For an Ethernet port for RoCE mode this field must be always valid and contain the remote GID address. For an Ethernet port for R-RoCE mode, this field contains IPv4 or IPv6 depending L3 version set in Address table entry pointed by sgid_index: when L3 type is IPV4 then rgid/rip bits [31:0] specify the remote IPv4 address and rgid/rip bits [127:32] are reserved, when L3 type is IPv6 then rgid/rip bits [127:0] contains an IPv6 address and if L3 type is GRH then rgid/rip bits [127:0] contains a GID address.	

8.10 Shared Receive Queue (SRQ)

A Shared Receive Queue enables associating multiple RQs with a single pool of receive buffers, allowing better memory utilization and management by pre-posting a receive buffer to a single pool that can serve multiple receive transports.

8.10.1 SRQs and QPs

When a QP is initialized, it can be associated with a SRQ. Association with an SRQ does not impact any send operations or responder operations that do not consume receive WQEs (RDMA-read, Atomic or RDMA-write without immediate). Upon reception of a WQE-consuming request on the RQ associated with SRQ, the WQE is dequeued from the shared list (SRQ) and this WQE controls the operation of the received request.

SRQ-attached RC QPs report invalid credits when generating Ack messages. This prevents remote requesters from getting into the limited state. However, if messages arrive while the SRQ has no receive descriptors available, an RNR Nak is sent on the QP (Reliable Connection) or the message is dropped (Unreliable Datagram). Correct provisioning of buffer pre-posting yields the best overall system performance, avoiding the RNR Nak flow (RC QPs) or dropping of packets (UD QPs).

Table 111 summarizes the major differences between SRQs and regular RQs.

Table 111 - Differences Between SRQ and QP

	QPs	SRQs
Receive WQE posting	To each QP RQ	To the SRQ
Number of WQEs pre-posted to the RQ n = number of QPs m = number of pre-posted descriptors required by the software application.	$n * m$	$\max(m, n)$
Memory resource requirement (B is the receive buffer size)	$n * m * B$	$\max(m, n) * B$
Completion reporting	All WQEs on a single QP report completion to the same CQ in order	WQEs on an SRQ report completion out of order to the CQs associated with QP.
Error semantics - WQE Flushing	When a QP gets into the error state all receive WQEs are flushed	When a QP gets into the error state, only WQEs in progress taken from the SRQ are flushed; other WQEs are kept intact on the SRQ.
SRQ Fatal Error	NA	If an SRQ becomes inaccessible, all QPs attached to the SRQ will move into the error state when they try to dequeue a WQE from the SRQ.
End to end credits reporting on receive side	Optional, controlled per QP	No reporting

8.10.2 SRQ Context

SRQ context structure is shown in Table 112.

Table 112 - SRQ Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset									
state				log_srq_size																												00h									
wq signature	cont srq		rl ky		log_rq_st ride				xrcd																								04h								
									page_offset								cqn																								08h
																																					0Ch				
													log_page_size																												10h
																																					14h				
																	pd																								18h
									lwm																wqe_cnt																1Ch
																																					20h-24h				
									dbr_addr																												28h-2Ch				
																																					30h-3Ch				

The properties of SRQ to be passed from software to hardware, are shown in Table 113.

Table 113 - SRQ Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	state	SRQ State 0000: good 0001: Error reserved on CREATE_SRQ	
	27:24	log_srq_ size	Log2 of the Number of WQEs in the RQ	

Table 113 - SRQ Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	31	wq_signature	If set, WQE signature will be checked on this SRQ.	
	30	cont_srq	WQEs on this SRQ must be contiguous on the WQ. link list field in the WQE is reserved	
	28	rlky	Reserved lkey enable	
	26:24	log_rq_stride	Log2 of stride on the shared RQ. Stride in bytes is $16 * (2^{\log\_rq\_stride})$ and the minimum SRQ stride allowed is 32 (that is, log_rq_stride must be ≥ 1). Choose the stride to be the lowest power of 2 satisfying the following inequality: <i>Max WQE size in RQ $\leq$ stride</i>	
	23:0	xrcd	The Scalable RC Domain. Messages coming to receive ports specifying this SRQ as RQ will be served only if XRC_Domain of the SRQ matches XRC_Domain of the transport QP of this message.	
08h	31:26	page_offset	Page offset in offset in quantas of (page_size / 64). For 4K pages, it is in 64B units.	
	23:0	cqn	CQ to report XRC messages directed to this SRQ	
10h	29:24	log_page_size	Log (base 2) of page size in units of 4KByte	
18h	23:0	pd	SRQ protection domain	
1Ch	31:16	lwm	Limit Water Mark - if the LWM is not zero, and the wqe_cnt drops below LWM when a WQE is dequeued from the SRQ, then an SRQ limit event is fired and the LWM is set to zero. Valid only upon QUERY_SRQ and CREATE_SRQ commands.	
	15:0	wqe_cnt	WQE count on the SRQ. Valid only upon QUERY_SRQ and CREATE_SRQ commands.	
28h-2Ch	64	dbr_addr	Physical address of DB Record See 8.9.2 Doorbell Record	

8.10.3 Creating and Destroying SRQs

The creation of an SRQ is similar to the creation of a QP. After the SRQ is created, WQEs can be posted by software and dequeued by hardware from the SRQ.

8.10.4 Attaching and Detaching QPs to/from SRQs

Attachment of QPs to an SRQ is performed on the RST2INIT_QP transition. On this transition, the driver should set the *srq* bit in the QPC and specify the SRQ number to be attached to in the

srqn field. Software is responsible to ensure that the SRQ is valid and is owned by hardware when attaching QPs to it.

Any transport service can be attached to SRQs. For InfiniBand transports, remote credits control must be set to invalid (QP.*ric* bit set to '1'). It is recommended to set the QP.*sic* in remote QPs that are attached to SRQs to speed up the wire negotiation of initial end-to-end credits.

Any transport service can be attached to SRQs, the SRQ has to be valid as long as QPs are attached to it. It is the responsibility of the driver to perform this and to ensure that no SRQs are destroyed while there are QPs attached to them.

A QP is automatically detached from an SRQ when it moves to the RST state. The QP has to be moved into the ERR state before moving to the RST state to avoid leakage of a WQE that is partly-consumed by the QP while it is moved from valid state to RST.

8.10.5 Posting WQEs to SRQ

Posting WQEs to an SRQ is described in detail in “Posting Work Request on Shared Receive Queue” on page 310. The DoorBell record for SRQ is identical to RQ DoorBell record and is described in “Doorbell Record” on page 308.

8.10.6 SRQ Completion Semantics

SRQ WQEs are dequeued from the SRQ and become assigned to certain QPs when a WQE-consuming message arrives on a QP attached to the SRQ. Completions are reported to the CQ associated with the respective QP receive transport (QP.*cqn_recv* field) except for XRC transport service, where CQ to report completions is specified in the SRQ. Note that SRQ WQEs may complete out of order and to different CQs. It is the responsibility of the driver to track the available space on the SRQ.

8.10.7 SRQ Limit Water Mark Notification

An SRQ can be configured to generate an event when the number of WQEs posted to the SRQ drops below a specified limit-watermark. Arming the Limit Water Mark notification mechanism is done by setting a value other than zero to the SRQ.lwm by the ARM_SRQ command (See “ARM_RQ” on page 2108). When the hardware de-queues a WQE from the SRQ and the SRQ.wqe_cnt drops below the SRQ.lwm, an SRQ Limit Event will be generated to the Event Queue as specified in Appendix 8.20.6, “Event Queue Mapping” (page 476). If when executing the ARM_SRQ command the SRQ.wqe_cnt is smaller than the new Limit Water Mark, an event will be generated.

8.10.8 Resizing SRQ

TBD

8.10.9 Contiguous SRQ

In a regular SRQ, receive WQEs may not be contiguous and are therefore connected using a link list. In a contiguous SRQ however, SW must post WQEs to SRQ in a contiguous manner and

HW will use this to perform performance optimizations. The link list pointers in the SRQ receive WQEs are reserved.

Contiguous SRQ is marked by the *cont_srq* bit in the SRQ context. See Table 112, “SRQ Context Layout,” on page 367.

[EMC] For contiguous SRQs - MSB of SRQ number must be 1. For non-contiguous SRQ, MSB of SRQ number must be 0.

8.10.10 SRQ Error Semantics

Error semantics on SRQs and QPs attached to SRQs are slightly different from those of regular QPs. If a QP enters the error state, the WQEs that have already been dequeued from the SRQ are completed with error or completely flushed. The rest of the WQEs are left intact in the SRQ and the SRQ state is not affected by the error.

If failure occurs while dequeuing a WQE, the SRQ moves to the error state and an affiliated asynchronous event is generated. While the SRQ is in the error state, QPs can be attached or detached from the SRQ and WQEs can still be posted. QPs that attempt to dequeue a WQE from the SRQ are moved to the error state by hardware.

To clean up an SRQ in the error state, SW has to destroy all QPs that are attached to the SRQ and then destroy the SRQ. After all resources have been destroyed, they can be reopened again, with the SRQ being opened first.

SRQ flushing semantics differ from regular QP flushing semantics. While regular QPs flush all WQEs that are posted after an error is introduced, SRQs tend to minimize the flushing. As they are shared, flushing WQEs can impact other QPs sharing the same SRQ. Therefore, if an error is detected on a QP attached to an SRQ, only WQEs that have already been dequeued from the SRQ for processing by the errant QP are flushed. The rest of the WQEs on the SRQ are unaffected by the error.

After SRQ flushing is complete, the ‘Last WQE Reached’ event (see “QP/SQ/RQ/SRQ Events” on page 480) is generated for the errant QP. (If no WQEs need to be flushed, this event is generated as the QP enters the error state.) Thus, no new completions are reported on the specific RQ until the QP state is modified. The event data includes the QP number of the errant QP (not the SRQ number).

8.11 XRC Shared Receive Queues

[Internal]TBD - add section of description like what we have in SRQ

8.11.1 XRC SRQ Context

XRC SRQ context structure is shown in Table 114.

Table 114 - XRC_SRQ Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
state				log_xrc_srq_size																											00h	

Table 114 - XRC_SRQ Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
				rl ky	basic cyclic rev wqe			log_rq_st ride		xrcd																				04h				
page_offset							dbr umem valid	cqn																				08h						
																																		0Ch
			log_page_size					user_index																										10h
																																		14h
								pd																										18h
lwm																wqe_cnt																		1Ch
																																		20h
dbr_umem_id																																		24h
dbr_addr																																		28h-2Ch
																																		30h-3Ch

The properties of XRC_SRQ to be passed from software to hardware, are shown in Table 115

Table 115 - XRC_SRQ Context Layout

Offset	Bits	Name	Description	Access
00h	31:28	state	SRQ State 0000: good 0001: Error Reserved on CREATE_SRQ	
	27:24	log_xrc_srq_size	Log2 of the number of WQEs.	
04h	31	wq_signature	If set, WQE signature is checked on this XRC_SRQ.	
	30	cont_srq	WQEs on this XRC_SRQ must be contiguous on the WQ. Link list field in the WQE is reserved.	
	29	must_not_use		
	28	rlky	Reserved lkey enable.	
	27	basic_cyclic_rcv_wqe	0: cyclic receive WQE always includes the 16 byte of ctrl (signature field). 1: cyclic receive WQE includes the 16 bytes of ctrl only when wq_signature==1. Valid only if HCA_CAP.basic_cyclic_rcv_wqe==1.	
	26:24	log_rq_stride	Log2 of stride on the shared RQ. Stride in bytes is $16 * (2^{\log\_rq\_stride})$ and the minimum XRC_SRQ stride allowed is 32 (that is, log_rq_stride must be ≥ 1). Choose the stride to be the lowest power of 2 satisfying the following inequality: <i>Max WQE size in RQ $\leq$ stride.</i>	
	23:0	xrcd	The Scalable RC domain. Messages coming to receive ports specifying this SRQ as an RQ will be served only if XRC_Domain of the SRQ matches XRC_Domain of the transport QP of this message.	
08h	31:26	page_offset	Page offset in offset in quantas of (page_size / 64). For 4K pages, it is in 64B units.	
	24	dbr_umem_valid	Indicates if dbr address is given directly or by by umem_id and umem_offset. When 0, dbr_addr allocates the physical address of the dbr. When 1, dbr_addr allocates the umem_offset of the dbr, and umem_offset is valid	
	23:0	cqn	CQ to report XRC messages directed to this XRC_SRQ.	

Table 115 - XRC_SRQ Context Layout (Continued)

Offset	Bits	Name	Description	Access
10h	29:24	log_page_size	Log (base 2) of page size in units of 4KByte.	
	23:0	user_index	User_index - an opaque identifier that software sets and which is reported to Completion Queue. 0xffffffff - user_index does not exist (e.g like cq_version==0). Reserved if HCA_CAP.cqe_version==0.	
18h	23:0	pd	SRQ protection domain.	
1Ch	31:16	lwm	Limit Water Mark - if the LWM is not zero, and the wqe_cnt drops below LWM when a WQE is dequeued from the XRC_SRQ, then an XRC_SRQ limit event is fired and the LWM is set to zero. Valid only upon QUERY_XRC_SRQ and CREATE_XRC_SRQ commands.	
	15:0	wqe_cnt	WQE count on the XRC_SRQ. Valid only upon QUERY_XRC_SRQ and CREATE_XRC_SRQ commands.	
24h	31:0	dbr_umem_id	UMEM ID of DB Record. Valid only when dbr_umem_valid==1. when set, dbr_addr indicates umem_offset instead of physical address of the DB Record.	
28h-2Ch	64	dbr_addr	Physical address of DB Record See 8.9.2 Doorbell Record	

8.12 Transport Interface Receive (TIR)

The transport interface receive (TIR) object is responsible for performing all transport related operations on the receive side. TIR performs the packet processing and reassembly and is also responsible for demultiplexing the packets into different RQs. Demultiplexing supports delivery of the packet into one or more RQs that are listed, or a hash based selection of a single RQ from a list (e.g. Receive Side Scaling).

TIR context is created through the CREATE_TIR command. The context can be then modified according to the TIR state machine through the MODIFY_TIR command. It can be queried through the QUERY_TIR command. When finalized, TIR context is destroyed through the DESTROY_TIR command.

The support of TIR and the number of supported TIRs is reported via the QUERY_HCA command through the *log_max_tir* parameter.

TIR Context format is shown in Table 116.

Table 116 - TIR Context Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
disp_type				nvmeotcp_en		tls_en																						04h				
																														08h		
																														0Ch		
				lro_timeout_period_usecs												lro_en- able_mas k				lro_max_msg_sz										10h		
								afu_id																						14h		
inline_rqn_vhca_id_valid																inline_rqn_vhca_id														18h		
				inline_ q_type				inline_rqn																						1Ch		

Table 116 - TIR Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											indirect_table	20h
rx_hash_symmetric		tunneled_offload_en																															
rx_hash_fn					self_lb_en		transport_domain																								24h		
rx_hash_toeplitz_key[319:0]																																28h-4Ch	
rx_hash_field_selector_outer (See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379)																																50h	
rx_hash_field_selector_inner (See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379)																																54h	
nvmeotcp_klm_list_table_mkey																																58h	
																																5Ch-ECh	

TIR Context fields descriptions are shown in Table 117.

Table 117 - TIR Context Fields

Offset	Bits	Name	Description
04h	31:28	disp_type	TIR Dispatching type mode 0x0: Direct - TIR delivers packets directly to the RQ/ VIR-TIO_NET_Q it is associated with, see inline_rqn. Receive hash function for this dispatcher type must be HASH_NONE. 0x1: Indirect- TIR delivers packets to RQ via Indirection table, see indirect_table field. TIR selects Indirection Table entry using RX hash value calculated on selected packet's fields, see rx_hash_field_selector fields. Receive hash function for this dispatcher type must not be HASH_NONE.
	27	tls_en	If set, TLS offload is supported. Supported only when HCA_CAP.tls==1. See Section 4.9, "TLS Offload," on page 216
	26	nvmeotcp_en	If set, NVMeoTCP offload is supported. Supported only when HCA_CAP.nvmeotcp==1.

Table 117 - TIR Context Fields

Offset	Bits	Name	Description
10h	27:12	lro_timeout_period_u-secs	Sets the LRO timer period value in usecs which will be used as LRO session expiration time for this TIR. Please note that if specified value is not supported by HW it will be adjusted to the nearest timeout period which is < requested timeout period. If requested timeout period is smaller than the minimal supported LRO will not work in this TIR. The supported LRO timeout periods can be queried via HCA_CAP.lro_timer_supported_periods[] as described in table Table 585 – “Per Protocol Networking Offload”. The LRO timer period value can be dynamically modified when required via MODIFY_TIR command. SW can query the actual LRO session expiration timeout value via QUERY_TIR (after CREATE_TIR or MODIFY_TIR are completed). Reserved when LRO is disabled on this TIR.
	11:8	lro_enable_mask	LRO enable bitmask Bit 0: IPv4_LRO - When set LRO for IPv4 is enabled Bit 1: IPv6_LRO - When set LRO for IPv6 is enabled Other - Reserved When bitmask is completely clear LRO is disabled on this TIR.
	7:0	lro_max_msg_sz	Sets the max size of coalesced segment in chunks of 256Bytes. When HCA_CAP.lro_max_message_size_mode is 0, this field sets max LRO IP payload size (TCP header + TCP payload). When HCA_CAP.lro_max_message_size_mode is 1, this field sets max LRO message size starting from L2 headers (L2 + L3 + TCP headers + TCP payload). 0 - disable LRO. Reserved when LRO is disabled on this TIR.
14h	23:0	afu_id	Acceleration function unit identifier. If set, then this TIR carries traffic to the Acceleration Function Unit identified by afu_id.
18h	31	inline_rqn_vh-ca_id_valid	If set, the inline_rqn_vhca_id field is valid ~~Supported only when HCA_CAP.m_pci_port==1. Supported only when HCA_CAP.m_pci_port 1 or when the TIR directly points to VIRTIO_NET_Q object created with full_emulation bit set.~~ ~~See Section 14.9, “Multi PCI Port,” on page 857~~
	15:0	inline_rqn_vhca_id	Specifies the vHCA of the RQ specified by the inline_rqn field- Valid only when inline_rqn_vhca_id_valid==1. See Section 14.9, “Multi PCI Port,” on page 857

Table 117 - TIR Context Fields

Offset	Bits	Name	Description
1Ch	26:24	inline_q_type	Inline queue type. 0x0: RQ - Table 124, “RQ Context Format,” on page 385 0x1: VIRTIO_NET_Q - See Section 8.27.18, “VIRTIO_NET_Q Object,” on page 571. Reserved for disp_type != direct.
	23:0	inline_rqn	Inline RQ number or VIRTIO_NET_Q number according to inline_q_type. Reserved for disp_type != direct.
20h	31	rx_hash_symmetric	Symmetric Hashing (reserved for rx_hash_fn HASH_NONE) When set the L3 and the L4 fields are sorted prior to insertion into the receive hash function. Reserved for HASH_NONE hash function.
	29	tunneled_offload_en	When set this TIR is enabled to support Stateless Offloads for tunneled packets. Note: If HCA_CAP.tunnel_cqe_offload_wo_tunnel_offload is set and driver parses CQE correctly according to CQE.tunneled field, this field should be set, even if stateless offloads are not supported for tunneled packets. Note: Configuration of each receive offload is described in Section 4.3.12, “Hypervisor Offloading for Tunneling,” on page 201 .
	23:0	indirect_table	Indirection Table ID. Reserved when disp_type != Indirect.
24h	31:28	rx_hash_fn	RX Hash Function 0x0: HASH_NONE - No Hash Function 0x1: HASH_INVERTED_XOR8 - Inverted XOR8 generates XOR8 results but with inverted LSBs. The number of inverted LSBs is dependent on the max RQT size. It is log base 2 of the RQT.rqt_max_size. 0x2: HASH_TOEPLITZ - Toeplitz Hash Function
	25:24	self_lb_en	Bitmask indicates which Self Loopback traffic to enable Bit 0: enable_unicast - supported only when HCA_CAP.self_lb_uc == 1 Bit 1: enable_multicast - supported only when HCA_CAP.self_lb_mc == 1 Must be 0 when tls_en == 1 or nvmeotcp_en == 1
	23:0	transport_domain	Transport Domain ID. Note that TIS should be created before creation of TIR which has the same transport domain, otherwise sw should modify self_lb_en in TIR to the same value
28h-4Ch	320	rx_hash_toeplitz_key[319:0]	Toeplitz hash key Reserved when rx_hash_fn is not HASH_TOEPLITZ

Table 117 - TIR Context Fields

Offset	Bits	Name	Description
50h	31:0	rx_hash_field_selector_outer (See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379)	This field sets which outer or the only packets headers fields should be selected for RX Hash. See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379. When this field is enabled both tunneling and non-tunneling packets can be delivered to that TIR. For tunneling packets this selector refers to the fields of the outermost headers while for non-tunneling packets the bitmask refers to the packets only headers. Note: that rx_hash_fields_selector_outer cannot be enabled together with rx_hash_fields_selector_inner. This means that if rx_hash_fields_selector_outer != 0 rx_hash_fields_selector_inner must be zero and vice versa. Reserved for rx_hash_fn HASH_NONE
54h	31:0	rx_hash_field_selector_inner (See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379)	This field sets which tunneled (inner) packets headers fields should be selected for RX Hash. See Table 118, “RX_HASH_FIELD_SELECT Structure Layout,” on page 379. When this field is enabled only tunneling packets can be delivered to that TIR. Only packets which contain the enabled fields can be delivered to that TIR, this must be done with proper Flow Table rules. For example if L4_SPORT is enabled and l4_protoco_type == TCP only tunneled TCP packets are allowed to reach this TIR. Note: that rx_hash_fields_selector_inner cannot be enabled together with rx_hash_fields_selector_outer. This means that if rx_hash_fields_selector_inner != 0 rx_hash_fields_selector_outer must be zero and vice versa. Note: rx_hash_fields_selector_inner can be enabled only when tunneled_offload_en == 1. Reserved for rx_hash_fn HASH_NONE
58h	31:0	nvmeotcp_klm_list_table_mkey	Mkey of the list of command capsule KLMs are located. The size of the table equal to the mkey size. Valid only when nvmeotcp_en==1

Table 118 describes the RX Hash selector which selects which packet fields are used for RX hash calculation that is used also for traffic spreading (RSS).

Table 118 - RX_HASH_FIELD_SELECT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
l3_prot_type		l4_prot_type		selected_fields																												00h

Table 119 - RX_HASH_FIELD_SELECT Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	l3_prot_type	This field sets the L3 protocol type (IPv4 or IPv6) SRC_IP and DST_IP fields refer to: 0: IPv4 1: IPv6 If both SRC_IP and DST_IP are disabled this field is ignored and it does not impact the RX hash behavior.	
	30	l4_prot_type	This field sets the L4 protocol type (TCP or UDP) L4_SPORT and L4_DPORT fields refer to: 0: TCP 1: UDP If both L4_SPORT and L4_DPORT are disabled this field is ignored and it does not impact the RX hash behavior.	
	29:0	selected_fields	Bitmask which sets which packets headers fields should be selected for RX Hash. Each bit represents a field as described below 0: SRC_IP 1: DST_IP 2: L4_SPORT 3: L4_DPORT 4: IPSEC_SPI 5-29: reserved Only packets which contain the enabled fields can be delivered to that TIR, this must be done with proper Flow Table rules. For example if L4_SPORT is enabled and l4_protocol_type == TCP only TCP packets are allowed to reach this TIR. Note that if this field refers to tunneling packets only tunneling packets with proper inner headers are allowed to reach this TIR.	

To modify a TIR, software provides a bitmask of the various fields that are being changed (bitmask not relevant to CREATE_TIR command). The following table summarizes the bitmasks and the supported state transitions where bits can be set.

Table 120 - MODIFY_TIR Bitmask

Bit	Field	CREATE	MODIFY	Comment
NONE	transport domain	R		
NONE	tls_en	R		

Table 120 - MODIFY_TIR Bitmask

Bit	Field	CREATE	MODIFY	Comment
0	LRO - lro_enable_mask - lro_timeout_period_usecs - lro_max_ip_payload_size	R	O	
1	indirect_table	R	O	
2	Hash - rx_hash_fn - rx_hash_symmetric - rx_hash_toeplitz_key - rx_hash_field_selector_outer - rx_hash_field_selector_inner	R	O	
3	tunneled_offload_en	R		
4	self_lb_en	R	O	Supported only if HCA_CAP.self_lb_en_modifiable==1

Legend

- R - Required parameter
- O - Optional parameter

8.13 Transport Interface Send (TIS)

The transport interface send (TIS) object is responsible for performing all transport related operations of the transmit side. Messages from Send Queues get segmented and transmitted by the TIS including all transport required implications, e.g. in the case of large send offload, the TIS is responsible for the segmentation.

TIS context is created through the CREATE_TIS command. The context can then be modified through the MODIFY_TIS command. It can be queried through the QUERY_TIS command. When finalized, TIS context is destroyed through the DESTROY_TIS command.

The support of TIS and the number of supported TISes is reported via the QUERY_HCA command through the *log_max_tis* parameter.

TIS Context format is shown in Table 121.

Table 121 - TIS Context Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
strict_lag_tx_port_affinity	tls_en			lag_tx-port_affinity					prio_or_sl																				00h			
																														04h-20h		
				transport_domain																										24h		
				underlay_qpn																										28h		
				pd																										2Ch		
																														30h-9Ch		

TIS Context fields descriptions are shown in Table 122.

Table 122 - TIS Context Fields

Offset	Bits	Name	Description
00h	31	strict_lag_tx_port_affinity	If set, the TIS is not subject to LAG TX port remapping and the native port will be used. See Section 14, “Virtualization,” on page 838.
	30	tls_en	If set, TLS offload is supported. Supported only when HCA_CAP.tls==1. Note that SQ must point to non-TLS enabled TIS. i.e. tls_en==0. See Section 4.9, “TLS Offload,” on page 216
	27:24	lag_tx_port_affinity	Defines the desired TX LAG port number. Valid only when LAG is created by CREATE_LAG command or when HCA_CAP.lag_tx_port_affinity==1. NA when strict_lag_tx_port_affinity==1. If 0 the device will assign the affinity. See Section 14, “Virtualization,” on page 838

Table 122 - TIS Context Fields

Offset	Bits	Name	Description
00h	19:16	prio_or_sl	For Ethernet, Ethernet Priority in prio[3:1] (bit 0 is reserved). For InfiniBand, InfiniBand Service Level.
24h	23:0	transport_domain	Transport Domain ID. Note that TIS should be created before creation of TIR which has the same transport domain, otherwise sw should modify self_l-b_en in TIR to the same value
28h	23:0	underlay_qpn	Underlay UD QPN. This field sets the QPN that is supposed to handle packets sent via this TIS. Valid only for IB. See Section 4.6.1.2, “Enhanced ULP Offloads,” on page 210.
2Ch	23:0	pd	Protection domain. valid only when tls_en i=1

To modify a TIS, software provides a bitmask of the various fields that are being changed (bitmask not relevant to CREATE_TIS command). The following table summarizes the bitmasks and the supported state transitions where bits can be set:

Table 123 - MODIFY_TIS Bitmask

Bit	Field	CREATE	MODIFY
0	prio	R	O
NONE	transport_domain	R	
NONE	tls_en	R	
NONE	underlay_qpn	R	
1	strict_lag_tx_port_affinity	R	O
2	lag_tx_port_affinity	R	O
NONE	pd	R	

Legend

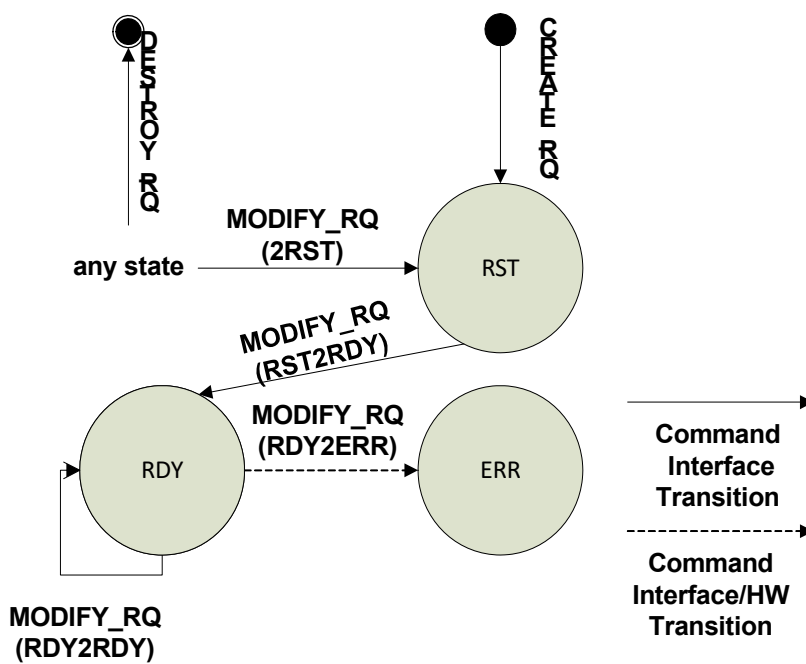
- R - Required parameter
- O - Optional parameter

8.14 Receive Queue (RQ)

The Receive Queue (RQ) object holds the destination for incoming packets/messages including certain offloads. RQs are attached to a TIR/Sniffer and report completions to their completion queue.

Multiple RQ types are supported: memory RQ type (which can be inline memory or RMP). Inline memory RQs include a work queue that is used to handle the incoming data memory delivery. RMP RQs do not have individual buffer to handle the incoming data but rather point to an RMP object, which can be shared across multiple RQs. The RMP object provides memory for handling incoming messages. RMP objects are described in detail in Section 8.17, “Receive Memory Pool (RMP),” on page 411. RQ context is subject to a well defined state transitions as illustrated in Figure 25.

Figure 25: RQ States



RQ context is created through the `CREATE_RQ` command. The context can be then modified according to the RQ state machine through the `MODIFY_RQ` command. It can be queried through the `QUERY_RQ` command. When finalized, RQ context is destroyed through the `DESTROY_RQ` command.

The support of RQs and the number of supported RQs is reported via the `QUERY_HCA` command through the `log_max_rq` parameter.

RQ Context format is shown in Table 124.

Table 124 - RQ Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset																				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										hairpin_offload_type	00h																			
vlan_strip_disable				mem_rq_type				state				flush_in_error_en				hairpin																																			
delay_drop_en																																																			
scatter_fcs																																																			
rllky																																																			
								user_index																				04h																							
								cqn																				08h																							
counter_set_id																												0Ch																							
								rmpn																				10h																							
dpp_wire_protocol								hairpin_peer_sq																				14h																							
dpp_segment_size												hairpin_peer_vhca																18h																							
dpp_mkey																														1Ch																					
dpp_buffer_size																														20h																					
																														24h-2Ch																					
wq (See Table 126, “Work Queue (WQ) Format,” on page 389)																														30h...																					

RQ Context fields descriptions are shown in Table 125.

Table 125 - RQ Context Fields

Offset	Bits	Name	Description
00h	31	rlky	Reserved LKey enable. When set the reserved LKey can be used on the RQ.
	30	delay_drop_en	If set, when Delay Drop state is active, a packet arriving to this RQ and does not have a free WQE will not be immediately dropped. See Section 8.14.3, “Drop-Less RQ,” on page 398.

Table 125 - RQ Context Fields

Offset	Bits	Name	Description
00h	29	scatter_fcs	If set, the FCS (Frame Check Sequence) field of a received Ethernet packet will be scattered into host memory along the packet payload. The FCS will be scattered as it appeared on the wire. That is, if the NIC manipulates the packet before scattering (adds/removes/changes headers), the FCS will not match the packet as it appears on the host memory. Supported only if HCA_CAP.scatter_fcs==1. Note that scatter_fcs should not be set for RQs connected to TIRs with LRO enabled. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
	28	vlan_strip_disable	VLAN Stripping Disable 0 - strip VLAN from incoming Ethernet frames 1 - do not strip VLAN from incoming Ethernet frames
	27:24	mem_rq_type	Memory RQ Type 0x0: MEMORY_RQ_INLINE - Inlined memory queue 0x1: MEMORY_RQ_RMP - RMP, RQ points to a remote memory pool. Supported only when HCA_CAP.mem_rq_rmp==1 0x2: DPP - supported only when HCA_CAP.dpp==1. when set all fields are reserved except of state, dpp_wire_protocol, dpp_mkey, dpp_mem_size and dpp_segment_size.
	23:20	state	RQ state 0x0: RST 0x1: RDY 0x3: ERR For QUERY_RQ - the current state is returned For MODIFY_RQ - the requested state is specified See Section 8.14.1, “RQ States Summary,” on page 397 for further details.
	18	flush_in_error_en	If set, and when RQ transitions into error state, the hardware will flush in error WQEs that were posted and WQEs that will be posted to an RQ. Otherwise, when RQ enters an error state, HW is not forced to flush in error all the WQEs, but still can generate completion. Reserved for mem_rq_type == MEMORY_RQ_RMP. For more details related to RQ Error behavior See Section 8.14.2, “RQ Error Semantics,” on page 398.
	17	hairpin	If set, this RQ is used for hairpin. See Section 4.8, “Hairpin,” on page 215.

Table 125 - RQ Context Fields

Offset	Bits	Name	Description
00h	3:0	hairpin_offload_type	Hairpin offload type. Valid only when hairpin==1. 0x0: HAIRPIN_ETH_RAW - supported only when HCA_CAP.hairpin==1 0x1: HAIRPIN_IB_RAW - supported only when HCA_CAP.hairpin_ib_raw==1 0x2: HAIRPIN_ETH2IPOIB - supported only when HCA_CAP.hairpin_eth2ipoib==1 0x3: HAIRPIN_IPOIB2ETH - supported only when HCA_CAP.hairpin_ipoib2eth==1
04h	23:0	user_index	User_index - an opaque identifier which software sets, which is reported to the Completion Queue. Reserved if HCA_CAP.cqe_version==0.
08h	23:0	cqn	Completion Queue Number.
0Ch	31:24	counter_set_id	Set of counters in which statistics on this RQ are collected. counter_set should be allocated using ALLOC_Q_COUNTER command. See Section 29.35, “Q Counters Commands,” on page 2185. 0x0 means not connected to a counter set.
10h	23:0	rmpn	RMP number. Reserved for non RMP RQs.
14h	31:24	dpp_wire_protocol	Direct Packet Placement protocol. The supported protocol reported in HCA_CA.dpp_wire_protocol. Valid only when rq_mem_type==DPP. 0x0: ST2110 0x1: ST2110_extended 0x2: GigEvision_EL0 0x3: GigEvision_EL1
	23:0	hairpin_peer_sq	Indicates the SQ peer of hairpin. Valid only when hairpin==1.
18h	31:16	dpp_segment_size	Direct Packet Placement segment size in granularity of Bytes. Must be between $2^{\text{HCA\_CAP.log\_min\_dpp\_segment\_size}}$ and $2^{\text{HCA\_CAP.log\_max\_dpp\_segment\_size}}$. Valid only when rq_mem_type==DPP.
	15:0	hairpin_peer_vhca	Indicates the vhca_id peer of hairpin. Valid only when hairpin==1.
1Ch	31:0	dpp_mkey	Direct Packet Placement Memory key of the buffer. Valid only when rq_mem_type==DPP.
20h	31:0	dpp_buffer_size	Direct Packet Placement Memory buffer size in granularity of dpp_segment_size. Must be $\leq$ from $2^{\text{HCA\_CAP.log\_max\_dpp\_buffer\_size}}$. Valid only when rq_mem_type==DPP.

Table 125 - RQ Context Fields

Offset	Bits	Name	Description
30h...	1600	wq (See Table 126, “Work Queue (WQ) Format,” on page 389)	Work Queue.

Table 126 summarizes the work queue fields and their applicability to send queues and receive queues.

Table 126 - Work Queue (WQ) Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h	
wq_type				wq_signature	end_padding_mode	cd_slave																					00h					
hds_skip_first_sge	log2_hds_buf_size										page_offset				lwm												04h					
									pd																				08h			
									uar_page																				0Ch			
	dbr_addr																												10h-14h			
	hw_counter																												18h			
	sw_counter																												1Ch			
													log_wq_stride								log_wq_pg_sz								log_wq_sz			

Table 126 - Work Queue (WQ) Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
dbr_umem_valid	wq_umem_valid		log_hairpin_num_packets									log_hairpin_data_sz								single_wqe_log_num_of_strides		two_byte_shift_en					single_stride_log_num_of_bytes		24h		
dbr_umem_id																														28h	
wq_umem_id																														2Ch	
wq_umem_offset																														30h-34h	
																														38h-BCh	
pas[0]																														C0h-C4h	
pas[1]																														C8h-CCh	
...																														...	

Work Queue fields descriptions are shown in Table 127.

Table 127 - Work Queue (WQ) Fields

Offset	Bits	Name	Description	SQ	RQ	RMP/XRQ
00h	31:28	wq_type	WQ type 0x0: WQ_LINKED_LIST - Linked List 0x1: WQ_CYCLIC - Cyclic Descriptors 0x2: LINKED_LIST_STRIDING_WQ - valid only when HCA_CAP.eth_striding_wq==1. The WQEs of the WQ will be linked in a list The WQE format will be as defined in “Shared Receive WQE format”. 0x3: CYCLIC_STRIDING_WQ - valid only when HCA_CAP.eth_striding_wq==1. The WQEs of the WQ will be consumed in order, in a cyclic manner The WQE format will be as defined in the “Shared Receive WQE Format” section. In this case (CYCLIC_STRIDING_WQ) the “next segment” of the WQE is reserved. Send Queue WQ must be set to 1.	v	v	v
	27	wq_signature	If set, WQE signature will be checked on this WQ.	v	v	v
	26:25	end_padding_mode	Scattering end of incoming send message (or raw Eth packet) 0: END_PAD_NONE - scatter as-is 1: END_PAD_ALIGN - pad to cache line alignment other reserved Note that padding does not go beyond the receive WQE scatter entry length. [internal] 2 - pad to cache line alignment beyond scatter entry if needed		v	v
	24	cd_slave	If set, WQ is a recipient of CD doorbells via the master SQ	v		
04h	31	hds_skip_first_sge	When set, the first scatter entry is skipped when HDS cannot be done. Reserved when HDS is disabled.		v	v
	30:28	log2_hds_buf_size	Log (base2) of HDS buffer size 0 - HDS disabled		v	v only for RMP

Table 127 - Work Queue (WQ) Fields

Offset	Bits	Name	Description	SQ	RQ	RMP/XRQ
04h	20:16	page_offset	Page offset in offset in quanta of (page_size / 64)	v	v	v
	15:0	lwm	Limit Water Mark when WQE count drops below this limit, an event is fired. 0- Disabled		v	v
08h	23:0	pd	Protection Domain. Used for accessing data through WQEs (scatter/gather).	v	v	v
0Ch	23:0	uar_page	UAR number allocated for ringing DoorBells for this WQ. For RQ this field is required only if cd_slave is enabled otherwise this field is reserved.		v	
10h-14h	64	dbr_addr	Physical address of DB Record See 8.9.2 Doorbell Record	v	v	v
18h	31:0	hw_counter	Current HW stride index Points to the next stride to be consumed by HW). Bits [31:16] are reserved. This field is available through the QUERY commands only. Otherwise reserved.	v	v	v
1Ch	31:0	sw_counter	Current SW WQ WQE index Points to the next stride to be produced by SW. Bits [31:16] are reserved. This field is available through the QUERY commands only. Otherwise reserved.	v	v	v
20h	19:16	log_wq_stride	The size of a WQ stride equals $2^{\text{log\_wq\_stride}}$.	v	v	v
	12:8	log_wq_pg_sz	Log (base 2) of page size in units of 4Kbyte	v	v	v
	4:0	log_wq_sz	When hairpin in SQ/RQ==0: WQ size in Bytes is $2^{(\text{log\_wq\_size} + \text{log\_wq\_stride})}$ and log_wq_sz must be less than HCA_CAP.log_max_wq_sz When hairpin in SQ/RQ==1: The WQ size in granularity of WQEs(packets) is $2^{\text{log\_wq\_size}}$ and log_wq_sz must be less than HCA_CAP.log_max_hairpin_wq_sz	v	v	v

Table 127 - Work Queue (WQ) Fields

Offset	Bits	Name	Description	SQ	RQ	RMP/XRQ
24h	31	dbr_umem_valid	Indicates if dbr address is given directly or by by umem_id and umem_offset. When 0, dbr_addr allocates the physical address of the dbr. When 1, dbr_addr allocates the umem_offset of the dbr, and umem_offset is valid	v	v	v
	30	wq_umem_valid	Indicates if wq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	v	v	v
	28:24	log_hairpin_num_packets	Log (base 2) of the number of packets of the hairpin RQ/SQ. Valid only when hairpin==1. Must be <= HCA_CAP.log_max_hairpin_num_packets	v	v	
	20:16	log_hairpin_data_sz	Log (base 2) of the wq size for hairpin RQ/SQ in granularity of Bytes. Valid only when hairpin==1. Must be <= HCA_CAP.log_max_hairpin_wq_data_sz	v	v	

Table 127 - Work Queue (WQ) Fields

Offset	Bits	Name	Description	SQ	RQ	RMP/XRQ
24h	11:8	single_wqe_log_num_of_strides	Number of strides in each WQE: $512 * 2^{\text{single\_wqe\_log\_num\_of\_strides}}$ Valid only when <code>wq_type == LINKED_LIST_STRIDING_WQ</code> or <code>CYCLIC_STRIDNG_WQ</code> . Values -6 to -1 representing 8 to 256 strides are accepted (as 4bit 2c representation) if <code>HCA_CAP.ext_stride_num_range</code> is set. See Section 8.9.1.3, “Striding Receive WQ,” on page 304		v	v
	7	two_byte_shift_en	When enabled, hardware pads 2 byte of zeros before writing the message to memory (e.g. for IP alignment). Valid only when <code>wq_type == LINKED_LIST_STRIDING_WQ</code> or <code>CYCLIC_STRIDNG_WQ</code> . See Section 8.9.1.3, “Striding Receive WQ,” on page 304		v	v
	2:0	single_stride_log_num_of_bytes	Stride size = $(2^{\text{single\_stride\_log\_num\_of\_bytes}}) * 64B$. Valid only when <code>wq_type == LINKED_LIST_STRIDING_WQ</code> or <code>CYCLIC_STRIDNG_WQ</code> . See Section 8.9.1.3, “Striding Receive WQ,” on page 304		v	v
28h	31:0	dbr_umem_id	UMEM ID of DB Record. Valid when <code>dbr_umem_valid==1</code> . when valid, <code>dbr_addr</code> indicates <code>umem_offset</code> instead of physical address of the DB Record.	v	v	v
2Ch	31:0	wq_umem_id	UMEM ID of the WQ. Valid only when <code>wq_umem_valid==1</code> .	v	v	v
30h-34h	64	wq_umem_offset	UMEM offset in the allocated UMEM for the WQ. Valid only when <code>wq_umem_valid==1</code> .	v	v	v
C0h-...	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)	Array of physical address structure (PAS) that map the work queue. Valid only when <code>wq_umem_valid==0</code>.	v	v	v

To modify an RQ, software provides a bitmask of the various fields that are being changed. Table 128 summarizes the bitmasks and the supported state transitions where bits can be set:

Table 128 - CREATE_RQ and MODIFY_RQ Bitmask

	Bit	Field	CREATE_RQ	RST2RDY	RDY2RDY	RDY2ERR	2RST
RQ	none	state	R	R	R	R	R
	2	scatter_fcs	R	O	O		
	none	rlky	R				
	none	cs	R				
	1	vsd	R	O	O		
	none	rq_type	R				
	none	rq_index	R				
	none	cqn	R				
	none	rmpn	R				
	none	flush_in_error_en	R				
	3	modify_rq_counter_set_id	R	O Only when HCA_ CAP. modi- fy_rq_ counte r_set_ id==1	O Only when HCA_ CAP. modi- fy_rq_ counte r_set_ id==1		
	none	delay_drop_en	R				
	none	hairpin	R				
	none	~~hairpin~~ hairpin_offload_type	R				
	none	hairpin_peer_vhca		R			
	none	hairpin_peer_rq		R			
WQ	none	wq_type	R				
	none	wq_signature	R				
	none	end_padding_mode	R				
	none		R				

Table 128 - CREATE_RQ and MODIFY_RQ Bitmask

	Bit	Field	CREATE_RQ	RST2RDY	RDY2RDY	RDY2ERR	2RST
	none	log_wq_pg_sz	R				
	none	log_wq_stride	R				
	none	log_wq_size	R				
	none	hds_skip_first_sge	R				
	none	log2_hds_buf_size	R				
	none	page_offset	R				
	0	lwm	R	O	O		
	none	pd	R				
	none	uar_page	N/A				
	none	dbr_addr	R				
	none	pas[...]	R				

Legend

- R - Required parameter
- O - Optional parameter

[\[TBD\] specify behavior in case of failure, bad init state, etc](#)

8.14.1 RQ States Summary

Table 129 summarizes the RQ states.

Table 129 - RQ States Summary

RMP	TIR Associated	RQ State	Description
None (RQ type is Direct) [TBD] Overlay	ALL	RST	Doorbells are not processed and WQEs are not flushed. Incoming messages are silently dropped or never reach the Receive Queue. Note: Prior to moving an RQ from RST into RDY state, the device driver has to initialize RQ DoorBell counter to zero.
	No	RDY	Work Requests can be submitted to the Receive Queue. Incoming messages never reach the RQ.
	YES	RDY	Work Requests can be submitted to the Receive Queue. Incoming messages are processed normally.
	ALL	ERR	If RQ.flush_in_error_en is enabled, Work Requests submitted to the Receive Queue are flushed in error. Otherwise, RQ is not forced to flush the posted Work Requests but Work Requests cannot be submitted to the Receive Queue and incoming messages are silently dropped or never reach the RQ. See Section 8.14.2, “RQ Error Semantics,” on page 398.
With RMP Memory RQ type is MEMO- RY_RQ_R MP RMP state is RDY	ALL	RST, ERR	Work Requests can be submitted to the corresponding RMP. Incoming messages are silently dropped or never reach the RQ.
	NO	RDY	Work Requests can be submitted to the corresponding RMP. Incoming messages never reach the RQ.
	YES	RDY	Work Requests can be submitted to the corresponding RMP. Incoming messages are processed normally.
With RMP. Memory RQ type is MEMO- RY_RQ_R MP RMP state is ERR	ALL	RST, ERR	Work Requests can be submitted to the corresponding RMP. Incoming messages are silently dropped or never reach the RQ.
	ALL	RDY	Work Requests can be submitted to the corresponding RMP. When an RQ tries to dequeue a WQE from the corresponding RMP, the RQ is placed in the ERR state and RQ Catastrophic Error event is generated. If not, the TIR that is connected to an RQ incoming messages never reach the RQ.

8.14.2 RQ Error Semantics

8.14.2.1 Memory RQ with Inline Memory Queue

A memory RQ with an inline receive memory queue (not connected to RMP) supports the following completion modes when it enters an error state:

- Flush in error WQEs that were posted and WQEs that will be posted to the RQ.
- When RQ enters Error state, it is not forced to flush in error all the WQEs but can still generate completions.

RQ.flush_in_error_en flags set the completion mode. For more details related to behavior of flush_in_error_en flag please refer to Table 124, “RQ Context Format,” on page 385.

8.14.2.2 RQ Associated with RMP

Error semantics on RQs attached to RMPs are slightly different from those of regular RQs. If an RQ enters the error state, the WQEs that have already been de-queued from the RMP are completed with error or completely flushed. The rest of the WQEs are left intact in the RMP and the RMP state is not affected by the error.

If failure occurs while de-queuing a WQE, the RMP moves to the error state and an affiliated asynchronous event is generated. While the RMP is in the error state, RQs can be attached or detached from the RMP and WQEs can still be posted. RQs that attempt to dequeue a WQE from the RMP are moved to the error state by hardware.

To clean up an RMP in error state, SW has to destroy all RQs that are attached to the RMP and then destroy the RMP. After all resources have been destroyed, they can be reopened again, with the RMP opened first.

RQs attached to RMP flushing semantics differ from regular RQs flushing semantics. While regular RQs can be configured to flush all WQEs that are posted after an error is introduced, RMPs tend to minimize the flushing. Since they are shared, flushing WQEs can impact other RQs sharing the same RMP. Therefore, if an error is detected on an RQ attached to an RMP, only WQEs that have already been de-queued from the RMP for processing by the errant RQ are flushed. The rest of the WQEs on the RMP are unaffected by the error.

After RMP flushing is complete, the ‘Last WQE Reached’ event (See Table 180, “Event_data Field - QP/SQ/RQ Events Layout,” on page 481) is generated for the errant RQ. (If no WQEs need to be flushed, this event is generated as the RQ enters the error state.) Thus, no new completions are reported on the specific RQ until the RQ state is modified.

8.14.3 Drop-Less RQ

For an Ethernet RQ, packets received when receive WQEs are exhausted are dropped. This behavior prevents slow or malicious entities at the host from affecting the network. In certain scenarios it may be preferred to configure the device to avoid such packet drops, assuming the posting of WQEs will resume shortly. This configuration is referred to as “Delay Drop” state.

Under such configuration, when a packet is received for an RQ with no receive WQEs, the packet processing is delayed, waiting for receive WQEs to be posted. Although this behavior causes the backpressure to be propagated to the network, it allows flow control mechanism to take over, thus avoiding packet loss. If receive WQEs are still not posted for a long period of

time, the packet is dropped and the Delay Drop state is exited. Future packets will not be delayed and will be dropped immediately. Timer expiration is indicated by the General Notification Event, “Delay Drop Timeout” subtype.

The Delay Drop is supported when `HCA_CAP.rq_delay_drop == 1`.

The feature is activated using the `SET_DELAY_DROP` command. The timeout value should be indicated by `delay_drop_timeout` field. Setting this value to the 0x0 disables the feature.

The Delay Drop may be reactivated by setting the timeout to a non-zero value.

The current state of the Delay Drop may be queried using the `QUERY_DELAY_DROP` command.

Only RQs enabled for Delay Drop will prevent packet drops when their WQEs are depleted. Marking an RQ to be drop-less is done by setting `delay_drop_en` in RQ Context using `CREATE_RQ` command.

It is recommended that Delay Drop would be used only when the port is configured to maintain flow control.

8.15 RQ Table (RQT)

RQT object holds a table of RQs. TIR points to RQT and uses it as indirection table. RQT allows to dispatch the packet only to single RQ from the table, accessed by table index.

RQT context is created through the `CREATE_RQT` command. The context can be then modified through the `MODIFY_RQT` command. It can be queried through the `QUERY_RQT` command. When finalized, RQT context is destroyed through the `DESTROY_RQT` command.

The support of RQT and the number of supported RQTs is reported via the `QUERY_HCA` command through the `log_max_rqt` parameter.

Table 130 - RQT Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h-10h
						list_q_type																									14h
rq_vhca_id format																															18h

Table 130 - RQT Context Format

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
																														1Ch-Ech		
rq_list_or_rq_vhca_list																														F0h...		

RQT Context fields descriptions are shown in Table 131.

Table 131 - RQT Context Fields

Offset	Bits	Name	Description
14h	26:24	list_q_type	List queue type. 0x0: RQ - Table 124, “RQ Context Format,” on page 385 0x1: VIRTIO_NET_Q - See Section 8.27.18, “VIRTIO_NET_Q Object,” on page 571.
	15:0	rqt_max_size	Sets the max allowed rqt_actual_size. If command attempts to set rqt_actual_size higher than rqt_max_size the command fails and RQT does not change. This field must be set according to HCA_CAP.log_max_rqt_size limitation, otherwise the command fails. rqt_max_size must be power of two.
18h	31	rq_vhca_id_format	If set, RQs-/VIRTIO_NET_Qs with vHCA IDs List is set, otherwise only RQ/VIRTIO_NET_Q list is set. ~~Supported only when HCA_CAP.m_pci_port==1.~~ Supported only when HCA_CAP.m_pci_port1 or when all of the VIRTIO_NET_Q list is created with full_emulation bit set See Section 14.9, “Multi PCI Port,” on page 857
18h	15:0	rqt_actual_size	The number of entries in this Receive Queues Table. rqt_actual_size must be power of two and <= rqt_max_size.
F0h-...	31:0	rq_list_or_rq_vhca_list	If rq_vhca_id_format ==0: Table of RQs. See Table 132, “RQ List Format,” on page 401 If rq_vhca_id_format ==1: Table of RQs with vHCA ID. See Table 134, “RQ vHCA List Format,” on page 401

Table 132 - RQ List Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
								rq_num[0]																				00h			
								rq_num[1]																				04h			
...																															

Table 133 - RQ List Fields

Offset	Bits	Name	Description
00h- ...	23:0	rq_num[...]	Table of RQs/ VIRTIO_NET_Q according to list_q_type . Note that the same RQN is allowed to appear multiple times. This table can be used as indirection table for RSS.

Table 134 - RQ vHCA List Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									
rq_vhca_entry[0] (Table 136, "RQ vHCA Entry Format," on page 402)																														00h-04h
rq_vhca_entry[1]																														08h-0Ch
...																														10h-..

Table 135 - RQ vHCA List Fields

Offset	Bits	Name	Description
00h- ...	64	rq_vhca_entry[...] (Table 136, “RQ vHCA Entry Format,” on page 402)	RQ vHCA Entry. See Table 136, “RQ vHCA Entry Format,” on page 402.

Table 136 - RQ vHCA Entry Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								rq_num																				00h				
																rq_vhca_id												04h				

Table 137 - RQ vHCA Entry Fields

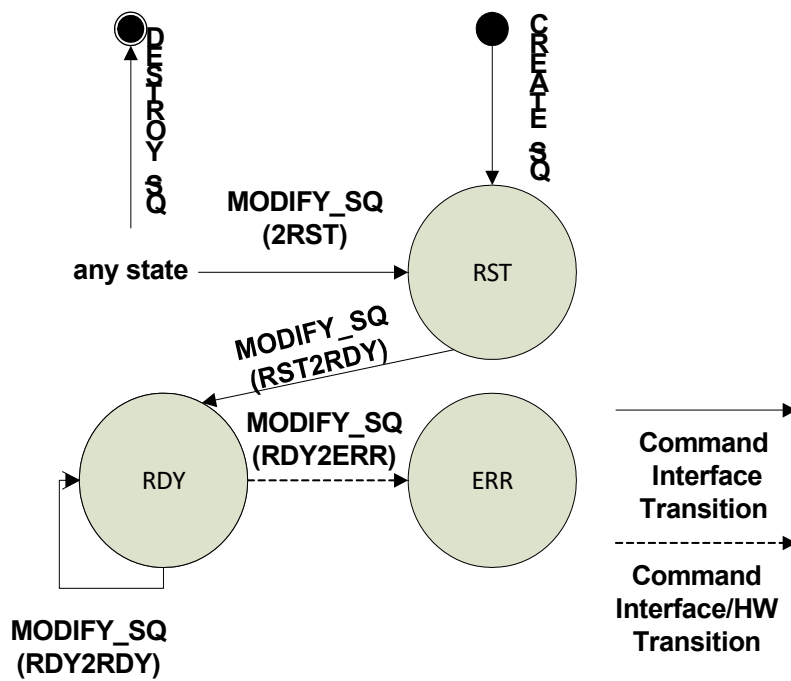
Offset	Bits	Name	Description
00h	23:0	rq_num	RQ Number/ VIRTIO_NET_Q according to list_q_type . Note that the same RQN is allowed to appear multiple times. This table can be used as indirection table for RSS.
04h	15:0	rq_vhca_id	vHCA ID of the RQ. Note that all vHCAs denoted by the different vhca_id array entries must have the same INIT_HCA.sw_owner_id and the same physical port GUID.

8.16 Send Queue (SQ)

The Send Queue (SQ) object holds the descriptor ring used to send outgoing messages and packets. The SQ is attached to a TIS and reports completions to its completion queue.

SQ context is subject to a well defined state transitions as illustrated in Figure 26.

Figure 26: SQ States



SQ context is created through the CREATE_SQ command. The context can be then modified according to the SQ state machine through the MODIFY_SQ command. It can be queried through the QUERY_SQ command. When finalized, SQ context is destroyed through the DESTROY_SQ command.

The support of SQs and the number of supported SQs is reported via the QUERY_HCA_CAP command through the *log_max_sq* parameter. SQs are typically associated with a single TIS. For support of SQ association into multiple TISes, check the QUERY_HCA_CAP *max_tis_per_sq*.

SQ Context fields descriptions are shown in Table 139.

Table 139 - SQ Context Fields

Offset	Bits	Name	Description
00h	31	rlky	Reserved LKey enable. When set the reserved LKey can be used on the SQ.
	30	cd_master	CORE direct master. When set, the SQ supports CORE direct WQEs.
	29	fre	Fast Register Enable. When set, the SQ supports Fast Register WQEs.
	28	flush_in_error_en	If set, and when SQ transitions into error state, the hardware will flush in error WQEs that were posted and WQEs that will be posted to SQ. Otherwise, when SQ enters an error state HW is not forced to flush in error all the WQEs. For more details related to SQ Error behavior refer to Section 8.16.2, “SQ Error Semantics,” on page 409.
	27	allow_multi_pkt_send_wqe	If set, posting multi packet send WQEs are allowed on this SQ. See Section 4.3.4, “Multi-Packet Send WQE Offload,” on page 191. This field can be set only if HCA_CAP.multi_pkt_send_wqe==1. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
	26:24	min_wqe_inline_mode	Sets the inline mode for the SQ. min_wqe_inline_mode defines the minimal required inline headers in Eth Segment of the SQ’s WQEs. If Eth Segment of the WQE does not contain the required inlined headers, the packet is silently dropped. Note: SQ.min_wqe_inline_mode must be >= Nic_vport.min_sq_wqe_inline_mode. See Table 29, “NIC_Vport Context Layout,” on page 287. For more details - See Section 8.9.4.1.5, “Eth Segment and Padding Segment,” on page 318.

Table 139 - SQ Context Fields

Offset	Bits	Name	Description
00h	23:20	state	SQ state 0x0: RST 0x1: RDY 0x3: ERR For QUERY_SQ - the current state is returned For MODIFY_SQ - the requested state is specified For further details See Section 8.16.1, “SQ States Summary,” on page 409.
	19	reg_umr	If set, this SQ is used only for posting UMR WQEs. Supported only when HCA_CAP.reg_umr_sq==1. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734.
	18	allow_swp	Allow software parsing. If set, SW can set swp_* fields in Eth segment. Supported only when HCA_CAP.Networking_Offloading_Capabilities.swp==1.
	17	hairpin	If set, this SQ is used for hairpin. The hairpin type is indicated in hairpin_offload_type field. See Section 4.8, “Hairpin,” on page 215.
	16	flow_table_update	When set, this queue is used for updating Flow Tables. for this type of SQ the following fields are reserved: Rlky, allow_multi_pkt_send_wqe, min_wqe_inline_mode, allow_swp, hairpin_peer_rq, hairpin_peer_vhca, cd_master, fre, reg_umr, hairpin, qos_para_vport_number, packet_pacing_rate_limit_index, tis_lst_sz, qos_queue_group_id, tis_num0
	3:0	hairpin_offload_type	Hairpin offload type. Valid only when hairpin==1. 0x0: HAIRPIN_ETH_RAW - supported only when HCA_CAP.hairpin==1 0x1: HAIRPIN_IB_RAW - supported only when HCA_CAP.hairpin_ib_raw==1 0x2: HAIRPIN_ETH2IPOIB - supported only when HCA_CAP.hairpin_eth2ipoib==1 0x3: HAIRPIN_IPOIB2ETH - supported only when HCA_CAP.hairpin_ipoib2eth==1
04h	23:0	user_index	User_index - an opaque identifier which software sets, which will be reported to the Completion Queue. Reserved if HCA_CAP.cqe_version==0.
08h	23:0	cqn	Completion Queue Number
0Ch	23:0	hairpin_peer_rq	Indicates the RQ peer of hairpin. Valid only when Hairpin==1.

Table 139 - SQ Context Fields

Offset	Bits	Name	Description
10h	15:0	hairpin_peer_vhca	Indicates the vhca_id of hairpin peer. Valid only when Hairpin==1.
14h	23:0	afu_id	Acceleration function unit identifier. If set, then this SQ carries traffic back from Acceleration function unit identified by afu_id.
1Ch	31:16	qos_para_vport_number	Identifier for the Para-VPort to which this SQ is associated. Value 0x0 indicates the SQ is not associated with a Para-VPort.
1Ch	15:0	packet_pacing_rate_limit_index	Index into Rate Limit table. 0x0 means there is no rate limitation. Supported when QoS_CAP==packet_pacing==1. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773.
20h	31:16	tis_lst_sz	The number of entries in the list of TISes. This is the list of Transport Interface Send (TISes) that are associated with this SQ.
	15:0	qos_queue_group_id	Handle of QUEUE_GROUP scheduling element representing the QoS group this SQ belongs to. Value 0x0 indicates this SQ does not belong to any specific group and will receive a device default QoS. Supported when QOS_CAP.nic_sq_scheduling is set.
2Ch	23:0	tis_num[0]	List of TIS numbers. Note: in this revision of PRM, only a single TIS is supported
30h...	1600	wq (See Table 126, “Work Queue (WQ) Format,” on page 389)	Work Queue

To modify an SQ, software provides a bitmask of the various fields that are being changed. Table 140 summarizes the bitmasks and the supported state transitions where bits can be set.

Table 140 - CREATE_SQ and MODIFY_SQ Bitmask

	Bit	Field	CREATE_SQ	RST2RDY	RDY2RDY	RDY2ERR	2RST
SQ	none	state	R	R	R	R	R
	none	rlky	R				
	none	cd_master	R				
	none	fre	R				
	none	sq_index	R				
	none	cqn	R				

Table 140 - CREATE_SQ and MODIFY_SQ Bitmask

	Bit	Field	CREATE_SQ	RST2RDY	RDY2RDY	RDY2ERR	2RST
	none	tis_lst_sz	R				
	none	tis_num[0]	R				
	none	flush_in_error_en	R				
	none	min_wqe_inline_mode	R				
	0	packet_pacing_rate_limit_index	R				
	none	reg_umr	R				
	1	qos_para_vport_number	R	O	O		
	none	hairpin	R				
	none	hairpin_hairpin_offload_type	R				
	none	hairpin_peer_rq	R	R			
	none	hairpin_peer_vhca	R	R			
	2	qos_queue_group_id	R	O	O		
	none	flow_table_update	R				
WQ	none	wq_type	R				
	none	wq_signature	R				
	none	end_padding_mode	N/A				
	none	cd_slave	R				
	none	log_wq_pg_sz	R				
	none	log_wq_stride	R				
	none	log_wq_size	R				
	none	hds_skip_first_sge	N/A				
	none	log2_hds_buf_size	N/A				
	none	page_offset	R				
	none	lwm	N/A				
	none	pd	R				
	none	uar_page	R				
	none	dbr_addr	R				

Table 140 - CREATE_SQ and MODIFY_SQ Bitmask

	Bit	Field	CREATE_SQ	RST2RDY	RDY2RDY	RDY2ERR	2RST
	none	pas[...]	R				

Legend

- R - Required parameter
 - O - Optional parameter
 - N/A - Not Applicable (irrelevant for SQs)
 - [\[TBD\] Looks like not all fields appear as required/optional, need to clean it up. Marked them as “R?” for now](#)
 - [\[TBD\] we must close whether we define DEFAULT values or force in CREATE_SQ to have all fields available and mask=0xFFFF...The fields to consider appear above in RR](#)
- [\[TBD\] specify behavior in case of failure, bad init state, etc](#)

8.16.1 SQ States Summary

Table 141 summarizes the SQ states.

Table 141 - SQ States Summary

SQ State	Description
RST	Doorbells are not processed and WQEs are not flushed. Note: Prior to moving an SQ from RST into RDY state, the device driver has to initialize SQ DoorBell counter to zero.
RDY	Work requests are processed normally.
ERR	If SQ.flush_in_error_en is enabled, Work Requests submitted to the SQ are flushed in error. Otherwise, an SQ is not forced to flush the posted Work Requests but Work Requests cannot be submitted to the Send Queue. See Section 8.16.2, “SQ Error Semantics,” on page 409.

8.16.2 SQ Error Semantics

SQ supports the following completion modes when it enters an error state:

- Flush in error WQEs that were posted and WQEs that will be posted to SQ.
- When SQ enters Error state it is not forced to flush in error all the WQEs.

SQ.flush_in_error_en flags set the completion mode. For more details related to the behavior of *flush_in_error_en* flag please refer to Table 138, “SQ Context Format,” on page 404.

8.16.3 Send WQE Inline Header

Each SQ's WQE must contain a minimal amount of packet headers inlined in the WQE's Eth Segment. Send WQEs with inline headers less than the required are dropped. The device supports different inline modes that are described in Section 8.16.3.1, "Inline Modes," on page 410, each inline mode requires a different minimal number of inline headers. The required inline modes can be queried by `HCA_CAP.wqe_inline_mode`.

HCA_CAP WQE inline modes:

- Mode 0 - The inline mode is L2.
- Mode 1 - The minimal required inline headers are set by *`SQ.min_wqe_inline_mode`*. Send WQEs with inline headers less than the required by *`SQ.min_wqe_inline_mode`* are dropped. The device supports different inline modes that are described in Section 8.16.3.1, "Inline Modes," on page 410. Each inline mode requires a different minimal number of inline headers. NIC Vport Context defines the minimal inline mode required for all SQs that belong to NIC Vport (See Section , "," on page 287). An attempt to create SQ with inline mode < minimal required by the Vport NIC will fail. SW must query the required minimal inline header by reading the NIC Vport Context field using `QUERY_NIC_VPORT_CONTEXT` (my Vport). On SQ creation, the NIC driver must set the minimal required inline header for the created SQ by setting *`SQ.min_wqe_inline_mode`* field. `SQ.min_wqe_inline_mode` must be $\geq$ NIC Vport Context *`min_sq_wqe_inline_mode`*. The e-switch manager (`HCA_CAP.eswitch_manager==1`) is responsible to configure minimal required inline header to vports belonging to its e-switch, depending on the match headers required in E-switch FDB in Ingress Acl Flow Tables. It should query the minimal required inline header mode for its VPort by calling `QUERY_NIC_VPORT_CONTEXT` (its Vport), and configuring other VPorts on e-switch so the inline mode for any e-switch Vport must be $\geq$ inline mode of e-switch manager VPort, i.e. the minimal inline mode of e-switch manager VPort is actually the minimal required inline mode for all Vports of the e-switch. This configuration is performed using `MODIFY_NIC_VPORT_CONTEXT` command.
- Mode 2 - No inline headers are required.

8.16.3.1 Inline Modes

- Mode 0: None - no inline headers are required.
- Mode 1: L2 - for LLC/SNAP frames, the packet inlined headers must include Max (17 Bytes, L2 headers till the last EtherType). For non LLC/SNAP frames, the packet inlined headers must include L2 headers till the last EtherType.
- Mode 2: IP - for packets that carry IP header, the packet inlined headers must include up to the end of the IP header. For packets that do not carry IP header, "L2" mode rules should be applied.

- Mode 3: TCP/UDP - for packets that carry TCP/UDP header, the packet inlined headers must include up to the end of the TCP/UDP header. For packets that do not carry TCP/UDP header, “IP” mode rules should be applied.
- Mode 4: reserved.
- Mode 5: Inner L2 - for tunneled packet, the packet inlined headers must include up to the end of the last inner EtherType. For non-tunneled packets, “TCP/UDP” mode rules should be applied.
- Mode 6: Inner IP - for tunneled packets that carry inner IP header, the packet inlined headers must include up to the end of the inner IP header. For packets that do not carry inner IP header, “inner L2” mode rules should be applied.
- Mode 7: Inner TCP/UDP - for tunneled packets that carry inner TCP/UDP header, the packet inlined headers must include up to the end of the inner TCP/UDP header. For packets that do not carry inner TCP/UDP header, “inner IP” mode rules should be applied.

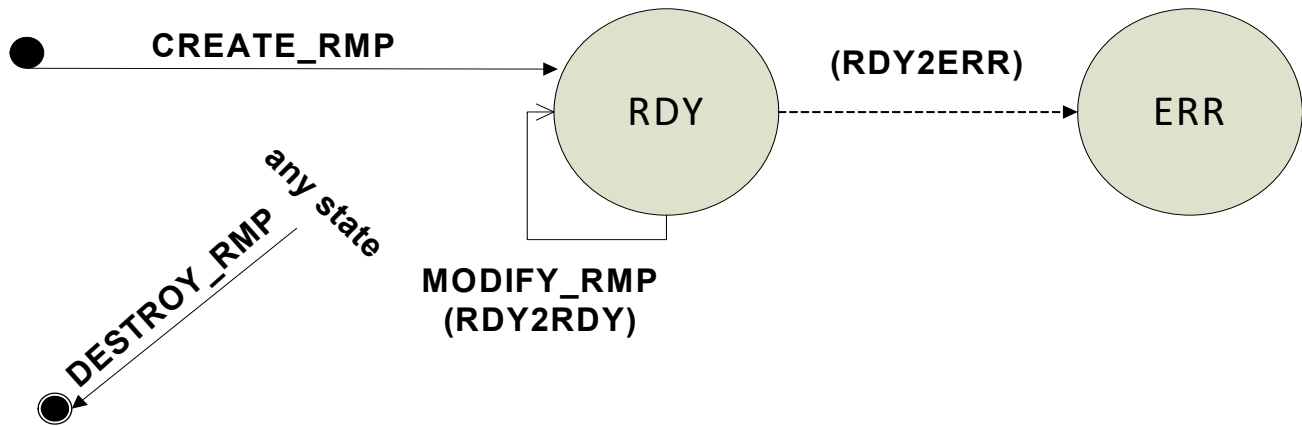
8.17 Receive Memory Pool (RMP)

The Receive Memory Pool (RMP) object holds the destination for incoming packets/messages that are routed to the RMP through RQs. RMP enables sharing of memory across multiple Receive Queues. Multiple Receive Queues can be attached to the same RQ and consume memory from that shared pool.

When using RMPs, completions are reported to the CQ pointed to by the RQ.

RMP context is subject to a well defined state transitions as illustrated in Figure 27.

Figure 27: RMP States



Legend:

- > **Command Interface Transition**
- - - - -> **Command Interface/HW Transition**

RMP context is created through the **CREATE_RMP** command. The context can be then modified according to the RMP state machine through the **MODIFY_RMP** command. It can be queried through the **QUERY_RMP** command. When finalized, RMP context is destroyed through the **DESTROY_RMP** command.

The support of RMP and the number of supported RMPs is reported via the **QUERY_HCA_CAP** command through the *log_max_rmp* parameter.

RMP Context format is shown in Table 142.

Table 142 - RMP Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h	
								state																								
basic_cyclic_rcv_wqe																													04h			
																													08h-1Ch			

Table 142 - RMP Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										30h...
wq (See Table 126, “Work Queue (WQ) Format,” on page 389)																															

RMP Context fields descriptions are shown in Table 143.

Table 143 - RMP Context Fields

Offset	Bits	Name	Description
00h	23:20	state	RMP state 0x1: RDY 0x3: ERR For QUERY_RMP - the current state is returned For MODIFY_RMP - the requested state is specified Must be RDY for CREATE_RMP For further information, See Section 8.17.1, “RMP States Summary,” on page 414
04h	31	basic_cyclic_rcv_wqe	0: cyclic receive WQE always includes the 16 byte of ctrl (signature field). 1: cyclic receive WQE includes the 16 bytes of ctrl only when wq_signature==1. Valid only if HCA_CAP.basic_cyclic_rcv_wqe==1 and wq.wq_type==WQ_CYCLIC
30h- ...	1600	wq (See Table 126, “Work Queue (WQ) Format,” on page 389)	Work Queue

To modify an RMP, software provides a bitmask of the various fields that are being changed. Table 144 summarizes the bitmasks and the supported state transitions where bits can be set.

Table 144 - CREATE_RMP and MODIFY_RMP Bitmask

	Bit	Field	CREATE_RMP	RDY2RDY
RMP	none	state	R	R
WQ	none	wq_type	R	
	none	wq_signature	R	
	none	end_padding_mode	R	

Table 144 - CREATE_RMP and MODIFY_RMP Bitmask

	Bit	Field	CREATE_RMP	RDY2RDY
	none	cd_slave	N/A	
	none	log_wq_pg_sz	R	
	none	log_wq_stride	R	
	none	log_wq_size	R	
	none	hds_skip_first_sge	R	
	none	log2_hds_buf_size	R	
	none	page_offset	R	
	0	lwm	R	O
	none	pd	R	
	none	uar_page	N/A	
	none	dbr_addr	R	
	none	pas[...]	R	

Legend

- R - Required parameter
- O - Optional parameter

[\[TBD\] specify behavior in case of failure, bad init state, etc](#)

8.17.1 RMP States Summary

RMP implements the two following states:

- RDY – this is the initial state. In this state software can post WQEs to the RMP. RMP provides WQEs to RQs attempting to consume WQEs normally.
- ERR – in this state software can post WQEs to the RMP. RMP does not provide WQEs to RQs attempting to consume WQEs; instead it causes the RQs to transition into the ERR state.

For further details on RMP states with correlation to RQ states, refer to Section 8.14.1, “RQ States Summary,” on page 397.

8.18 Flow Table

Packet processing by the device requires classifying them into flows. Each flow may have a different processing path and may lead to a different destination. Packet classification is done using the Flow Table mechanism.

8.18.1 Position in Processing Flow

Flow tables are used in several stages of packet processing flow. Each stage has a unique set of Flow Tables. The following processing stages using Flow Tables are defined:

- NIC Receive
- NIC Receive RDMA
- NIC Receive Sniffer
- NIC Transmit
- NIC Transmit RDMA
- NIC Transmit Sniffer
- E-Switch Forwarding Data Base (FDB)
- E-Switch Ingress Access Control List (ACL)
- E-Switch Egress Access Control List (ACL)

Each Flow Table type of the above list may support a subset of the Flow Table capabilities and may have special characteristics and limitations described in “Characteristics of Flow Table Types”.

8.18.2 General Structure

The Flow Table is built as a ternary content-addressable memory (TCAM). Each entry in the Flow Table represents a Flow by defining a match criteria and a match value. Packets processed by the Flow table attempt to match every entry, in order, starting at entry 0. The first entry that matches the packet defines the processing actions that should be immediately applied to the packet. The matching Flow is considered as ‘hit’ by the packet.

8.18.2.1 Match Criteria

The Flow Table entry match criteria is defined by a set of masks. Each mask represents one of three types of parameters:

- Packet header field - fields derived from parsing the packet:
 - MAC header - Source Address, Destination Address, Ethertype
 - VLAN header - VLAN ID, Priority (PCP), CFI/EDI
 - **EMD tag - Embedded Meta Data tag, defined by Ethertype 0x8CE4, with 6 bytes of data**
 - IPv6(v4) header - Source Address, Destination Address, Traffic Class (Type Of Service), Next Header (Protocol) [TBD - flow label ? DSCP/ECN ?](#)
 - TCP header - Source Port, Destination Port, Flags
 - UDP header - Source Port, Destination Port, Flags
 - BTH - Destination QP
 - VXLAN header - VNI
 - GRE header - Key, Protocol

- For supported encapsulation packets MAC, VLAN, TCP and UDP headers for the inner packet
- **Packet header logic - fields derived from logic on packet headers**
 - **VLAN tag - TBD**
 - **IP Fragment - TBD**
- **Packet meta-data field - fields derived from external source**
 - Source VPort
 - Source QP Number
 - metadata_reg_a
 - Metadata_reg_c_0 – metadata_reg_c_7

TBD - IB fields

Set bit in a mask means the appropriate bit in the appropriate field in the match value should be compared to the corresponding bit from the processed packet. Cleared bit means the appropriate bit in the appropriate field in the match value is ignored.

[Internal] hit logic is: $\text{hit} = (\text{mask} \& \text{packet fields}) == (\text{mask} \& \text{flow fields})$

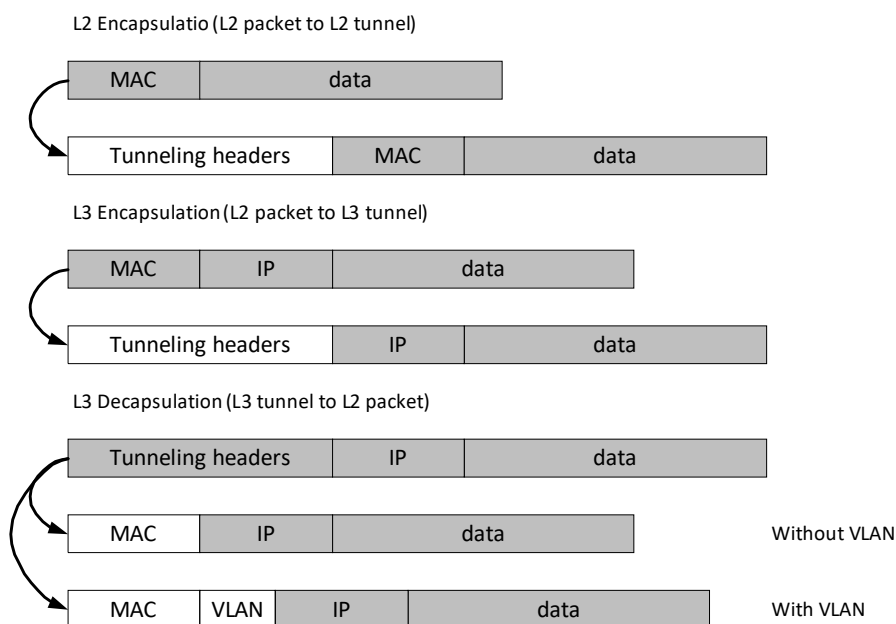
8.18.2.2 Processing Actions

When a packet matches a Flow, the actions specified for the Flow are applied. The possible actions for a Flow are as follows:

- *Drop* - Packet processing is stopped
- *Forward* - Packet processing continues in one or more processing entities specified in the Flow Table entry.
- *Allow* - Packet processing continues in a well-known pre-defined manner.
- *Count* - increment a packet counter, and add the packet size to a byte counter.
- *Reformat* - Change the packet format by adding or removing some of its headers. Figure 28 describes the supported reformat types (grey headers are taken from the original packet)
- *Decapsulate* - Remove outer packet headers of one of the supported encapsulating packet types.
- *Modify Headers* - Modify the value of the packet headers
 - *Set* - Set a header field to a certain value
 - *Add* - Add a certain value to a header field
 - *Copy* - Copy a header field into another header field
- *Pop VLAN* - Remove the VLAN tag immediately following the MAC header.
- *Push VLAN* - Add a VLAN tag immediately following the MAC header.
- *IPsec Decrypt* - Perform IPSEC Decryption offload for the packet, including transforming cypher data into plaintext data

- IPsec Encrypt - Perform IPSEC Encryption offload for the packet, including transforming plaintext data into cypher data
- **FPGA Vendor Acceleration - Send packet to FPGA Vendor Acceleration.**

Figure 28: Packet Reformat Types



8.18.2.3 Flow Table Chaining

Packet classification may require a hierarchical approach. For that purpose, a Flow with a *Forward* action may specify another Flow Table as the next processing entity for the packet. The destination Flow Table must be of the same type (i.e. belongs to the same processing stage). A packet matching this flow will continue to search for another match in the destination Flow Table starting at the first Flow. For every Flow Table type there is a single root table.

Chaining Flow Tables must never create loops.

8.18.2.4 Multi-Processing Paths

When the same packet needs to be processed multiple times, a Flow with a *Forward* action may indicate multiple destination processing entities. A packet matching this flow will be processed at each processing entity specified in the Flow separately. This mechanism may be used for multicasting the packet to several destinations.

Multi-processing paths to multiple Flow Tables must never be nested.

8.18.2.5 Default Behavior

Each Flow Table must be explicitly allocated before it may be used. In the absence of the root Flow Table in a given processing stage, a default behavior is applied for all packets going through this stage. The same default behavior is applied for a packet that matches a Flow with *Default* action or a packet that does not match no any Flow.

Each Flow Table Type has a defined default behavior described in “Characteristics of Flow Table Types”. This behavior can be overridden by allocating the root Flow Table, and defining a Flow with an empty match criteria and the required action.

Another method to redefine packet behavior when it does not match any of the Flows in the table, is to explicitly define a miss behavior in `table_miss_action`. One miss behavior could be a different table in the same steering domain (support is indicated by `identified_miss_table`) and the other could be moving to the root of an alternative table domain. The alternative table domain is specific for each Table Type (support is indicated by `table_miss_action_domain`)

8.18.2.6 Invalid Flows

Prior to the definition of a Flow in a Flow Table or after deleting a Flow, the Flow is Invalid. Packets never ‘hit’ Invalid Flows. If during lookup on a certain Flow Table an Invalid Flow is encountered, the packet simply moves on to the next Flow in the table in attempt to find a match.

8.18.2.7 Packet Classification Ambiguities

The match criteria of a single Flow may include any number of fields. It is the responsibility of the programmer to assure that once packet classification is done, all fields that define the packet class were matched, and no ambiguity remains.

For example, matching IP destination address does not guarantee that the packet is an IP packet. The Flow (or any of the Flows leading to it through Flow Table Chaining) must match the Ether-type field of the packet to assure an IP packet of the correct version.

In addition, a flow should not match or attempt to modify (*Modify Header* action, *Pop VLAN* action, *Reformat*) non-existing fields of a packet, as defined by the packet classification process. For example, matching destination IP address is not allowed for flows matching Ethertype to a non-IP value, or flows not matching the Ethertype at all.

Since packet classification process may span multiple flows in multiple Flow Tables, packet classification ambiguities or non-existing field matching should be avoided by the programmer. Failing to do so may result in undefined behavior, either failing the flow configuration or causing errors in packet classification.

8.18.2.8 Flow Statistics

Each Flow may collect statistics of the traffic matching the flow. The statistics are gathered in a flow counter, collecting the number of packets matching the flow and their total Byte count.

8.18.2.9 Flow Tagging

Each Flow may be marked with a tag, the attached packet matching the Flow, and exposed in the Completion report (CQE) *flow_tag* field. The tag could be used to identify the Flow through which the packet was processed. The CQE will reflect last tag encountered during packet processing. An unmarked Flow, will not modify the current tag attached to the packet.

Note: Unless `nic_rx_flow_tag_multipath_en` is set, A Flow tag should not be set for Flows beyond Multi-Processing split, if a Flow tag was already set prior to the split. The value of the tag in the CQE is not defined in these cases.

8.18.2.10 MetaData Registers

As part of Packet processing, metadata can be carried with the packet, used to store previous decisions, or used as auxiliary storage. The metadata registers are of several types, each with a slightly different behavior:

- **Type A** – this register is used in the transmit steering tables, and is loaded with the value of `flow_table_metadata` field in the Ethernet Segment of the WQE. The supported width of the register is indicated in `metadata_reg_a_width` field in `QUERY_HCA_CAP`.

[Internal] this register goes through loopback

- **Type B** – this register is used in the receive steering tables, and its value is passed to `flow_table_metadata` field in the CQE.

[Internal] matching for this field is not supported before ConnectX-6 Lx.

- **Type C** – a set of 32-bit registers - can be used for matching and Header Modify operations.

8.18.2.10.1 MetaData Between Table Types

Registers may or may not hold their value when crossing table type domains. If the registers hold any information that may be privileged only to the entity setting it, they should be cleared by the entity before allowing packet processing to continue. On the other hand, Metadata registers should not be assumed to have a known initial value, unless specified otherwise (e.g. Type A registers).

If Metadata is required to pass from the FDB to VPort tables (e.g. switch to Switch Manager), this information can be carried by Register C. The supported registers are indicated by `QUERY_HCA_CAP.fdb_to_vport_reg_c_id`, and the supported Vports can be queried by `ESW_VPORT_CONTEXT.fdb_to_vport_reg_c`. To ensure the required registers are passed, `fdb_to_vport_reg_c_id` should be set in `ESW_VPORT_CONTEXT.fdb_to_vport_reg_c_id`.

Note: Passing information from FDB to Vport table is only supported for non-tunneled packets.

8.18.2.11 Special Flow Tables

While basic structure of the Flow Tables is based on TCAM, the device supports special tables, allowing advanced steering options.

8.18.2.11.1 Hash Split Tables

This type of table allows for selecting the next processing stage of the packet by hashing the packet headers or packet metadata. Common use case is an implementation of a load balancer, where multiple flows should be mapped evenly to multiple handling queues, while keeping all packets of the same flow consistently mapped to the same queue. Flows in hash split tables

implicitly match the hash result, therefore there is no meaning to the match value of the Flow, and Flows are inserted by their bucket index.

For configuration details, see Section 8.18.3.4.1, “Hash Split Group,” on page 422 and Section 8.18.3.9.3, “Hash Split Flow,” on page 426.

8.18.3 Configuration Interface

8.18.3.1 Allocating a Flow Table

A new Flow Table is allocated using the “CREATE_FLOW_TABLE - Allocate a New Flow Table” command. The processing stage that the Flow Table belongs to is indicated in *table_type* field. The support for each of the Flow Table types is indicated by *ft_support* field in “Flow Table NIC Capabilities Layout”, under the appropriate *flow_table_properties_<type>*. When applicable the Vport holding the Flow table is indicated in *Vport* field.

The number of Flows in the table is indicated in *log_size* field. The table size must not exceed *log_max_ft_size* in “Flow Table NIC Capabilities Layout”, under the appropriate *flow_table_properties_<type>*. The number of allocated Flow Tables must not exceed *log_max_ft_num* in the same capabilities structure.

A Flow table that is used for Decapsulation should be indicated by *decap_en*. A Flow table that is used for Packet *Reformat* should be indicated by *reformat_en*. The support for such tables is indicated by *decap* and *reformat* fields respectively in the Flow Table type Capabilities.

8.18.3.2 Selecting the Root Flow Table

The root Flow Table for each Flow Table Type is defined, by default, as the Flow Table with level 0. Only a single Flow Table with level 0 is allowed. The exceptions are overlay NIC RX/TX Flow Table used for ULP offloads (see Section 8.18.3.2.1, “Selecting the Root Flow Table for ULP,” on page 420).

Explicitly setting or replacing the root Flow Table is allowed by using SET_FLOW_TABLE_ROOT command. The new root Flow Table should be allocated prior to execution of the command. The previous root table (if exist) is not deleted. Its relevant resources need to be released explicitly if no longer needed. The support for setting the root table is indicated by HCA_CAP.modify_root.

ACL Flow Tables have a different Root per Vport. The FDB Flow Table can be either managed as a single Root for all Vports, or as Root per Vport. The support for the latter is indicated by QUERY_HCA_CAP.sw_fdb_flow_table_root_per_port field. If driver uses this model, it must set this field by SET_HCA_CAP.

8.18.3.2.1 Selecting the Root Flow Table for ULP

For ULP, a separate “overlay” root Flow Table can be created per underlay QP. That allows to manage a separate and independent Flow Tables hierarchy for each overlay network.

Note: This mean that multiple Flow Tables with level 0 are allowed with a restriction: a single level 0 Flow Table per root Flow Table.

Note: Each underlay QP can be associated with a single root Flow Table, while each root Flow Table can be shared between multiple QPs.

8.18.3.3 Flow Table Level

When using an hierarchal approach for packet classification, the different hierarchies (levels) must be explicitly defined at Flow Table allocation. Chaining Flow tables must be done so a Flow in a Flow table of a certain level may only point to a Flow Table of a **higher** level. Note that if QUERY_HCA_CAP.ignore_flow_level is set, this limitation may be ignored by setting ignore_flow_level in SET_FLOW_TABLE_ENTRY.

When using Multi-Processing Paths, a Flow pointing to multiple Flow Tables or multiple TIRs must follow these guidelines:

- The Flow table containing this Flow must have a level **lower** than 64.
- The Flow tables pointed by this Flow (not relevant if the Flow points only to TIRs) must have a level **equal or higher** than 64. This does not apply to the last flow table in the destination list when HCA_CAP.nic_rx_multi_path_tirs = 1.

The Flow Table level is defined at allocation time by **level** field in “CREATE_FLOW_TABLE - Allocate a New Flow Table”. It must not exceed **max_ft_level** in “Flow Table NIC Capabilities Layout”, under the appropriate **flow_table_properties_<type>**.

8.18.3.3.1 Flow Table Identifier

Upon completion of the Flow Table allocation, a handle is returned in **table_id** field. This handle is unique for the specific Flow table type and Vport (when applicable).

8.18.3.4 Allocating a Match Criteria - Flow Groups

The match criteria of a Flow is defined before setting the actual Flow. A set of consecutive Flows in a certain Flow Table with the same match criteria is considered a Flow Groups. Flow Groups are allocated after the relevant Flow Table is allocated using the “CREATE_FLOW_GROUP - Define a New Flow Group” command.

The Flow Table a Flow Group belongs to is identified by the **table_type**, **table_id** and **vport** (when applicable) fields.

The range of Flows belonging to the Flow Group is defined by **start_flow_index** and **end_flow_index** fields. A Flow Group could be as little as a single Flow. The Flow Groups must not exceed the size of the Flow Table, and must not overlap. a Flow may only be associated with a single Flow Group.

The match criteria is given in **match_criteria** field, which is the set of masks as defined in “Match Criteria”. Each bit in **match_criteria_enable** field validates a sub-set of the **match_criteria**.

Each mask in the **match_criteria** may have one of the following values:

- FULL match - the corresponding field is compared completely, all bits of the mask are set.

- NONE match - the corresponding field is completely ignored, all bits of the mask are cleared.
- PARTIAL match - some of the bits of the corresponding field are compared.

A NONE match is supported for all fields in the *match_criteria*. A FULL match for a field is supported if the corresponding bit of *ft_field_support* in “Flow Table NIC Capabilities Layout” is set. a PARTIAL match is supported if the corresponding bit of *ft_field_bitmask_support* in “Flow Table NIC Capabilities Layout” is set, both under the appropriate *flow_table_properties_<type>*.

Note: The MAC header field destination address (*dmac*) mask 01:00:00:00:00:00 is also supported if a FULL match is supported for this field.

8.18.3.4.1 Hash Split Group

To implement a hash split, a Flow Group should be created with *group_type* HASH_SPLIT. The size of this group defined by *start_flow_index* and *end_flow_index* fields must be a power of 2. To define the fields and mask used for the hash calculation, a Match Definer object must be used, instead of the *match_criteria* field. Hash split tables must always have power of 2 number of rule. This type of table is supported when *HCA_CAP.flow_group_type_hash_split==1*.

8.18.3.4.2 Flow Group Identifier

Upon completion of the Flow Group allocation, a handle is returned in *group_id* field. This handle is unique for the specific Flow table the Flow Group is associated with.

8.18.3.5 Allocating a Flow Counter

A new flow counter is allocated using the “ALLOC_FLOW_COUNTER - Allocate a Flow Counter” command. The command returns the *flow_counter_id* handle to a future reference of the flow counter.

The support for flow counters in each of the Flow Table types is indicated by *flow_counter* field in Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742, under the appropriate *flow_table_properties_<type>*.

The number of available flow counters is indicated by *max_flow_counter* in Table 1571, “HCA Capabilities Layout,” on page 1671.

Each flow counter can only be assigned to Flows in Flow Tables of a single type. Distributing the flow counters between the different Flow Table types (for example between the eswitch and NIC tables), is outside the scope of this document and left for driver implementation.

Note: Flow Counters are supported only for NIC Rx, RDMA Rx, Egress ACLs and Sniffer Flow Tables.

8.18.3.6 Allocating a Packet Reformat Context

A flow performing *Reformat* action, requires a list of headers to be added to the packet. These headers are placed in a context allocated and populated by ALLOC_PACKET_REFORMAT_CONTEXT command. The type of reformatting should be specified in *reformat_type*, and the size of the headers should be specified in *reformat_data_size*. The size cannot exceed *max_encap_header_size*. The actual header is placed in *reformat_data* as a list of bytes.

A handle *packet_reformat_id* is returned for reference in the flow definition.

When multiple destinations require receiving a different version of the packet, it may be achieved by multiple-processing paths. However, sometimes, this mechanism may not be available.

The device supports attaching encapsulation types of reformat to the destination Uplink in the FDB Flow Table. The support is indicated by *Log_max_fdb_encap_uplink* field in QUERY_HCA_CAP.

In order to attach the reformat operation to the destination, the extended destination entries should be used (for all destinations and counters) in the flow_context by setting Extended_destination_entry field. The relevant destination entry should have destination_id indicating Uplink, the packet_reformat field should be set, and the packet_reformat_id inside the destination list entry should hold the handle of the Packet Reformat Context.

A packet already encapsulated by the device should not be directed to Encapsulate destination. Specifically Encapsulate action is not supported in a flow while a destination list entry indicating another encapsulation.

8.18.3.7 Encapsulation Entropy

When using the device packet encapsulation offload, the device calculates an entropy value, representing the inner packet headers. The entropy field is placed inside the outer packet headers. For UDP-type encapsulations, the entropy is placed in the source port field of the UDP header. For GRE-type encapsulations, the entropy is placed in the 8 LSB of the key field in the GRE header. If the device does not recognize the encapsulation type, the entropy is not placed in the packet.

Entropy setting can be controlled using PCMR register (See Table 1167, “PCMR - Ports Check Mask Register Layout,” on page 1357). If encapsulation offload is not used, force_entropy_cap should be set to 0x0.

Entropy setting is enabled/disabled using entropy_calc, and could be additionally enabled/disabled for GRE encapsulation by entropy_gre_calc.

Note: If both ports use GRE encapsulation, entropy_gre_calc setting should be the same for both ports.

8.18.3.8 Allocating a Header Modification Context

A flow performing *Modify Headers* action, requires a context indicating which headers to modify, and what kind of modification should be performed. The context is allocated and populated by ALLOC_MODIFY_HEADER_CONTEXT command. The type of table for which the context is created should be specified in *table_type*. The number of different modifications is specified in *num_of_actions*. The number of actions must not exceed *max_modify_header_actions* according to the Flow Table type Capabilities.

Note: IP header modifications are not allowed for packets whose upper layer headers are affected by the modification, e.g. (ICMPv6 checksum). For TCP and UDP packets IP header modifications are allowed.

Note: TTL field in IPv4 header should not be modified if CQE checksum field is used, unless HCA_CAP.ttl_checksum_correction==1.

8.18.3.9 Adding a Flow

A new Flow is defined using the “SET_FLOW_TABLE_ENTRY - Set Flow Table Entry” command.

The Flow Table a Flow belongs to is identified by the *table_type*, *table_id* and *vport* (when applicable) fields. The Flow Group the Flow belongs to is identified by the *group_id* field.

The location of the Flow inside the Flow Table is defined by the *flow_index* field. The location must not exceed the range of Flows defined by the Flow Group.

The match value is given in *match_value* field of the *flow_context* structure. The relevant fields inside the match value are defined by the match criteria associated with the Flow Group. Any other field is reserved and should be set to 0x0. For the FDB Flow Table, Flows which are known to be applicable only to packets arriving from specific origin, this information should be indicated by *flow_source* field. providing this information may save device resources (supported if QUERY_HCA_CAP.*flow_spurce* is set).

The Flow Tag attached to packets matching the Flow is indicated in *flow_tag* field. A value of 0x0 indicates an unmarked Flow, which does not modify the Flow Tag already attached to the packet.

Note: The processing order of identical Flows (with the same match value) in the same Flow Group is not guaranteed.

The total number of valid Flows in all Flow Tables of a single type must not exceed *log_max_flow* in the same capabilities structure.

8.18.3.9.1 Flow Actions

The field *action* holds a list of actions to be applied if a packet matches the Flow, as defined in “Processing Actions”. Some actions may be set independently, and some are mutually-exclusive.

- *Drop*, *Default* and *Forward* actions are mutually-exclusive. One of them must be applied to every Flow.
- *Reformat* and *Decapsulate* actions are mutually-exclusive.
- *Reformat* and *Decapsulate* actions cannot be set in conjunction with Forwarding the packet to a Flow Table destination unless HCA_CAP.reformat_and_fwd_to_table is set. In any case, programmer should not attempt to Reformat/Decapsulate the same packet more than once.
- *Drop* action is mutually-exclusive with any other action, except for Count action
- *Reformat* and *Modify Header* are mutually-exclusive unless HCA_CAP.reformat_and_modify_action is set. For L3 decapsulation *Reformat*, *Pop VLAN* action is not supported
- *Pop VLAN* and *Push VLAN* actions may not be set with the following actions:
 - *Drop*
 - *Decapsulate/Reformat*, unless HCA_CAP.reformat_and_vlan_action is set. For L3 decapsulation *Pop VLAN* is not supported.
 - *Modify Headers*, unless HCA_CAP.modify_and_vlan_action is set

- Forward to another Flow table, unless vlan and fwd_to table is set in QUERY_HCA - CAP. In any case programmer should not attempt to push vlan or pop vlan for the same packet more than once.
- IPsec Encrypt and IPsec Decrypt are mutually-exclusive.
- *Fpga Vendor Acceleration action must be set with Allow or Forward actions.*

When multiple actions are set for the same Flow (if applicable), the order of their application is not defined, except for:

- Modify Headers action is performed prior to packet encapsulation action (Reformat), and after packet decapsulation action (Reformat) .
- Pop Vlan is performed after a packet decapsulation action
- Push Vlan is performed prior to packet encapsulation action.
- IPsec Encrypt and Decrypt are performed after all packet modifications

When a Flow has a *Forward* action, the number of processing entities the packet should be destined to is specified in *destination_list_size*. Each destination is defined by a handle *destination_id* and a type *destination_type*. The defined destination types are:

- Flow Table
- TIR
- VPort
- QP
- VHCA_RX
- VHCA_TX

When specifying multiple destinations, the order between them is not defined. The destination list must be comprised of destinations of the same type. The number of destinations in a single Flow must not exceed *log_max_destination* in “Flow Table NIC Capabilities Layout”, under the appropriate *flow_table_properties_<type>*.

Note: Each TIR may only be used in a single type of Flow Table. Adding flows, in Flow Tables of different types and forwarding to the same TIR is not allowed. Only when *HCA - CAP.allow_sniffer_and_nic_rx_shared_tir==1*, then the same TIR can be used by multiple Flow Tables of the following different types: NIC receive, receive sniffer, transmit sniffer. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.

8.18.3.9.2Flow Counters

Flow counters can be associated with one or more flow(s) to collect statistics on the monitored flow(s).

Each flow counter set includes a number of counters as described in Table 1554 - QUERY_-FLOW_COUNTER Output Structure Layout

Before associating a *flow_counter_set* to a flow, a flow counter set must be allocated using `ALLOC_FLOW_COUNTER`.

The `ALLOC_FLOW_COUNTER` return *counter_set_id* which can be used as a handle for attaching the flow counter set to flow(s), query of the flow counter set and deallocating of the flow counter set.

Upon allocation, all counters within the counter set are set to 0.

`DEALLOC_FLOW_COUNTER` deallocates a *flow_counter_set*. Deallocating of a flow counter set which has flow(s) associated with it will fail. A counter which was deallocated cannot be read.

`QUERY_FLOW_COUNTER` command is used to query and reset a counter set specified by the *counter_set_id*.

The number of destinations in a single flow must not exceed *log_max_flow_counter* in Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742, under the appropriate *flow_table_properties_<type>*.

8.18.3.9 Hash Split Flow

Adding a Flow to a hash split group means deciding the required actions according to the hash result. When adding the Flow using `SET_FLOW_TABLE_ENTRY` command, the device will add an implicit match on the hash result, making the *match_value* field reserved. The *flow_index* field is used to determine the bucket number (hash result modulo group size), where the Flow would be inserted.

8.18.3.10 Redefining a Flow

An existing flow may be modified. Flow modification is guaranteed to be atomic. That is, once the flow is modified, new packets will not act according to the old flow definition. The modified flow must hold all requirements and restrictions applicable for any flow.

Note: For e-Switch FDB Flow Table, packets received from the uplink may not see the flow modification at the same time as packets received from other Vports.

The support for modifying a flow is indicated by *modify_flow_en* field in `QUERY_HCA_CAP` (Table 1587, “Flow Table Properties Layout,” on page 1753) for the appropriate Flow Table type. Modifying a flow is done using the `SET_FLOW_TABLE_ENTRY` command (See Table 2146, “SET_FLOW_TABLE_ENTRY Input Structure Layout,” on page 2023), using *opmod* value *Modify*.

The supported modifications are for:

- Action
- *flow_tag*
- Destination list (size and members)
- Flow Counter list (size and members)
- Packet Reformat Context identifier
- Modify Header Identifier

- VLAN tag for *Push Vlan* action

The modified fields should be indicated in the bitmask ***modify_enable***. The new values are taken from the appropriate fields in the ***flow_context***. Other fields in this context should match the original flow definition. When adding or removing a *Forward/Count* action, the destination/flow counter list must be modified as well.

8.18.3.11 Freeing Resources

Flows are invalidated using the “DELETE_FLOW_TABLE_ENTRY - Invalidate Flow Table Entry” command. The Flow Table a Flow belongs to is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. The Flow Group the Flow belongs to is identified by the ***group_id*** field. The location of the Flow inside the Flow Table is defined by the ***flow_index*** field. A Flow must be valid (previously added) to be invalidated.

Flow Groups are freed using the “DESTROY_FLOW_GROUP - De-allocate a Flow Group” command. The Flow Table a Flow Group belongs to is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. The Flow Group is identified by the ***group_id*** field. A Flow Group can only be freed if it was previously allocated and all the Flows associated with it are Invalid.

Flow Tables are freed using the “DESTROY_FLOW_TABLE - De-allocate a Flow Table” command. The Flow Table is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. A Flow Table can only be freed if it was previously allocated and all the Flow Groups associated with it are freed.

Flow Counters are freed using the “DEALLOC_FLOW_COUNTER - De-Allocate Flow Counter” command. The flow counter is identified by the ***flow_counter_id*** field. A flow counter can only be freed if it was previously allocated and all the flows associated with it are freed.

8.18.3.12 Querying Device Database

Flows may be queried using the “QUERY_FLOW_TABLE_ENTRY - Query Flow Table Entry” command. The Flow Table a Flow belongs to is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. The Flow Group the Flow belongs to is identified by the ***group_id*** field. The location of the Flow inside the Flow Table is defined by the ***flow_index*** field. A Flow must be valid (previously added) to be queried.

Flow Groups may be queried using the “QUERY_FLOW_GROUP - Query Flow Group” command. The Flow Table a Flow Group belongs to is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. The Flow Group is identified by the ***group_id*** field. A Flow Group must be allocated to be queried.

Flow Tables may be queried using the “QUERY_FLOW_TABLE - Query Flow Table” command. The Flow Table is identified by the ***table_type***, ***table_id*** and ***vport*** (when applicable) fields. A Flow Table must be allocated to be queried.

Flow Counters may be queried using the “QUERY_FLOW_COUNTER - Query Flow Counter” command. The Flow Counter is identified by the ***flow_counter_id*** field. A Flow Counter must be allocated to be queried. When querying the Flow Counter for the statistic of a Flow, the counter may be atomically reset, by setting the ***clear*** field.

8.18.3.13 Special Flows Definition

8.18.3.13.1 Prio-Tagged and Untagged Packets

A Flow matching only prio-tagged packets should include the ***vlan_tag*** field and the ***first_vid*** field in the match criteria. In addition, the Flow should match them to 0x1 and 0x0 respectively. A Flow matching only untagged packets should include the ***vlan_tag*** field in the match criteria, and the Flow should match it to 0x0. To define the same behavior for both types of packets, a single flow may be used instead. This Flow should include the ***vlan_tag*** field and the ***first_vid*** field in the match criteria, and the flow should match them to 0x0 and 0x0 respectively.

8.18.4 Characteristics of Flow Table Types

8.18.4.1 NIC Receive

Default Behavior - *Drop packet*

Supported Destination types - *Flow Table, TIR*

Special limitations:

- The field ***source_sqn*** of the match criteria is not supported
- The field ***source_vport*** of the match criteria is supported only for `vport_group_manager(HCA_CAP.vport_group_manager==1)`.
- Multiple processing paths are supported for multiple Flow Tables.
In case of `HCA_CAP.nic_rx_multi_path_tirs = 1`, multiple processing paths are supported also for multiple TIRs. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
In case of `HCA_CAP.nic_rx_multi_path_tirs_fts = 1`, multiple processing paths are supported also for single/multiple TIRs followed by single/multiple Flow Tables. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.
The last Flow Table in a destination list is not required to be of a level equal or greater than 64 like other Flow Tables in the destination list, if `HCA_CAP.nic_rx_multi_path_tirs = 1`.
- *Push VLAN* and *Pop VLAN* actions are not supported
- Action *Count* and ***flow_tag*** for the same flow are not supported, unless ***nic_rx_flow_tag_count_en*** is set in Table 1583, “Flow Table NIC Capabilities Layout”
- Action Modify Headers and ***flow_tag*** for the same Flow are not supported.

8.18.4.2 NIC Receive RDMA

Default Behavior - *RDMA Transport Offload*

Supported Destination types - *Flow Table, QP and VHCA_RX*.

The alternative steering Table for this table is the NIC Rx Table

Special limitations:

- The fields *flow_tag* and *source_sqn* of the match criteria are not supported.
- Action *Count* is allowed only for Flows matching either UC or MC packets, but not both (destination_mac bit 40 must be part of the match).
- The table is valid only when nic_vport_ctx.roce_en is equal to 1.
- Multiple processing paths to QP destinations are not supported
- *Push VLAN* and *Pop VLAN* actions are not supported
- Destination TIR is allowed only if HCA_CAP.nic_rx_rdma_fwd_tir is set

8.18.4.3 NIC Receive Sniffer

TBD

Default Behavior - *Drop packet*

Supported Destination types - *Flow Table, TIR*

Special limitations:

- The fields *source_spn* and *source_vport* of the match criteria are not supported
- Multiple processing paths are not supported

8.18.4.4 NIC Transmit

Default Behavior - *Forward packet to NIC Vport (Transmit)*

Supported Destination types - *Flow Table*

Special limitations:

- The field *source_vport* of the match criteria are not supported
- Multiple processing paths are not supported
- *flow_tag* not supported.
- *Push VLAN* and *Pop VLAN* actions are not supported

8.18.4.5 NIC Transmit RDMA

Default Behavior - *Transmit packet*

Supported Destination types - *Flow Table and VHCA_TX.*

Special limitations:

- The fields *source_vport* of the match criteria are not supported
- Multiple processing paths are not supported
- *flow_tag* not supported.
- Action Count is allowed only for Flows matching either UC or MC packets, but not both (destination_mac bit 40 must be part of the match).
- The table is valid only when nic_vport_ctx.roce_en is equal to 1

- *Push VLAN* and *Pop VLAN* actions are not supported

8.18.4.6 NIC Transmit Sniffer

TBD

Default Behavior - *Drop packet*

Supported Destination types - *Flow Table, TIR*

Special limitations:

- the fields *source_spn* and *source_vport* of the match criteria are not supported
- Multiple processing paths are not supported

8.18.4.7 E-Switch Forwarding Data Base (FDB)

Default Behavior

For packets from the Uplink - Forward to E-switch manager vport(HCA_CAP.eswitch_manager==1)

For packets transmitted from NIC VPorts - Forward to the Uplink

Supported Destination types - *Flow Table, VPort, TIR (See limitation note below)*

Special limitations:

- Flow Counters are supported under the following restrictions:
 - A Flow may have Count action only if the match criteria includes the full destination MAC address, or *extended_fdb_flow_counter_en* is set and the match criteria includes bit 40 of the destination MAC address, or *fdb_flow_counter_en*[0] is set.
 - A Flow may have *Count* action only if the match criteria includes the full source VPort, or *fdb_flow_counter_en*[1] is set.
 - A Flow may have *Count* and *Forward* actions only if destination list includes VPort(s) only, or *fdb_flow_counter_en*[1] is set.
 - The same Flow Counter may not be used in multiple Flows, unless they match the same destination MAC address and source VPort, or *fdb_flow_counter_en*[1] is set.
- Multiple processing paths are supported for multiple VPorts.
- Multiple processing paths are supported VPort and Flow Table if all of the following conditions are met:
 - QUERY_HCA_CAP.fdb_multi_path_to_table==1.
 - A destination of type Flow Table only appears once, as the last destination in the list.(the Flow Table is not required to have a level equal to or greater than 64 as usually required.) Flow Tables marked as termination (FLOW_TABLE_CONTEXT.termination_table==1) may appear more than once, at any place in the destination list. Support for such tables is indicated by FLOW_TABLE_PROPERTIES.termination_table==1.
 - The Flow match criteria includes the full source VPort.

- The matched source VPort is the Uplink, or there is only a single VPort in the destination list.
- flow_tag not supported.
- *Reformat* action that reformats the packet to a tunneled packet is supported only for Flows that forward the packet to the Uplink.
- *Decapsulate* action and *Reformat* action that reformats a tunneled packet to a non-tunneled packet, are supported only for Flows that forward the packet from the Uplink.
- Enabling Reformat or Decapsulate actions for a Flow table using reformat_en or decap_en must be done when all VFs are down. If QUERY_HCA_CAP.multi_fdb_encap is set, this requirement is applicable only if all other FDB Flow Tables do not enable these actions.
- *Push VLAN* action is supported under the following restrictions:
 - The action cannot be added concurrently to this table and E-Switch Ingress (ACL) table.
 - The Flow must have the Uplink and only the Uplink as the destination. If HCA_CAP.vlan_and_fwd_to_table==1, a Flow Table destination is also allowed. In any case programmer should not attempt to push vlan for the same packet more than once.
- *Pop VLAN* action is supported under the following restrictions:
 - The action cannot be added concurrently to this table and E-Switch Egress (ACL) table.
 - The Flow must not have the Uplink as its destination. If HCA_CAP.vlan_and_fwd_to_table==1, a Flow Table destination is also allowed. In any case programmer should not attempt to pop vlan for the same packet more than once
- TIR is supported as a destination only if HCA_CAP.fdb_single_tir_destination is set. The TIR can be used as the only destination of the flow, or in addition to a Flow Table destination.
- A Flow may modify the IPv4 TTL only if the packet is exclusively destined to the uplink, or if HCA_CAP.esw_fdb_ipv4_ttl_modify=1.
- Modify Headers action is supported only if the packet is forwarded to a Vport. If HCA_CAP.esw_fdb_modify_header_fwd_to_table is set, the action is also supported when the packet is forwarded to another Flow Table.
-
-

8.18.4.8 E-Switch Ingress Access Control List (ACL)

Default Behavior - *Forward the Packet to FDB*

Supported Destination types - *Flow Table and Vport*

Special limitations:

- The field *source_vport* of the match criteria is not supported.

- Multiple processing paths are supported if all of the following conditions are met:
 - QUERY_HCA_CAP.ingress_acl_forward_to_vport is set.
 - A destination of type Flow Table only appears once, as the last destination in the list.(the Flow Table is not required to have a level equal to or greater than 64 as usually required.)
- flow_tag not supported.
- flow counters are not supported.
- This table is only applicable after an FDB table is created.
- Pop VLAN action is not supported
- Push VLAN action cannot be added concurrently to this table and FDB table.

8.18.4.9 E-Switch Egress Access Control List (ACL)

- Default Behavior - *Forward the Packet to NIC VPort (Receive)*
- Supported Destination types - *Flow Table, Vport.*
- Special limitations:
 - Multiple processing paths are supported if all of the following conditions are met:
 - HCA_CAP.egress_acl_forward_to_vport is set ==1.
 - A destination of type Flow Table only appears once, as the last destination in the list.(the Flow Table is not required to have a level equal to or greater than 64 as usually required).
 - Vport destination is supported if HCA_CAP.egress_acl_forward_to_vport==1. Vport must not be the Uplink.
 - flow_tag not supported.
 - This table is only applicable after an FDB table is created.
 - Push VLAN action is not supported
 - Pop VLAN action cannot be added concurrently to this table and FDB table.

8.19 Completion Queues

HCA implements Completion Queues used to post completion reports upon completion of Work Request. This section discusses CQ structure and operation.

A CQ is an object containing the following entities:

1. **Completion Queue Buffer** - A virtually-contiguous circular buffer accessible by user-level software and used by hardware to post completions - writing Completion Queue Elements (CQEs). This area must be allocated when creating the CQ. Maximum supported CQ size can be retrieved by the QUERY_HCA_CAP command (*log_max_cq_sz*). CQE size is 64 or 128 bytes and is configured while creating the CQ.
2. **Completion Queue Context (CQC)** - The data structure describing the CQ properties that is passed to the device by SW while creating the CQ. HCA hardware can support up to 16M CQs per virtual HCA; the actual number supported by the given HW configuration can be

retrieved by the QUERY_HCA_CAP command (*log_max_cqs*). For more details see “[Completion Queue Context \(CQC\)](#)” on page 716.

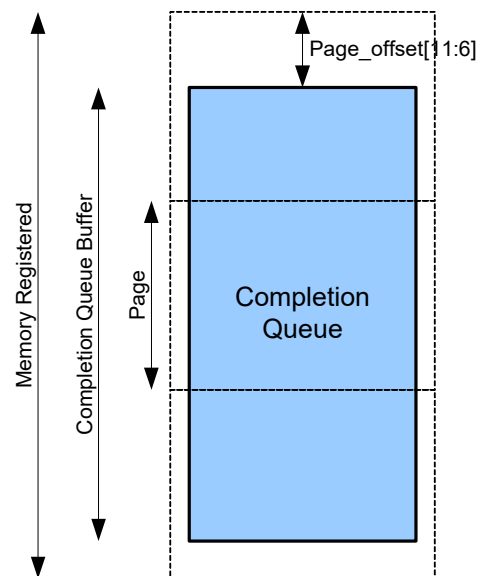
3. **CQ DoorBell Record** - A structure containing information of recently-posted CQ commands and consumer index and is accessible by user-level software.

8.19.1 Completion Queue Buffer

A CQ is a virtually-contiguous memory buffer used by the HCA to post completion reports to and used by application software to poll completion reports from. This buffer is allocated by the host software at CQ creation and its physical address is passed to the HCA. The memory associated with this buffer is passed to the HCA as a list of physical pages. CQ buffer must be aligned on CQE-size boundary¹.

Each CQ buffer contain two objects - CQ and CQ Doorbell Record. The CQ buffer structure is shown in Figure 29.

Figure 29: Completion Queue Buffer



Hardware posts completions (writing CQEs) to the CQ. Upon CQEs consumptions, SW updates the CQ Doorbell Record. In addition to the CQE consumption information, CQ Doorbell Record contains information about the CQ status (Armed for Event, Armed for Solicited Event).

CQ is a virtually-contiguous circular buffer that contains Completion Queue Entries (CQEs) posted by the HCA upon completion of I/O operation(s) associated with this CQ. Application (non-privileged) software can read the CQ buffer to poll completions posted by the HCA. The buffer is managed by two counters:

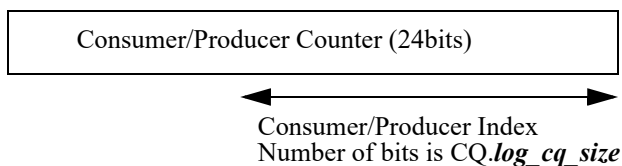
- *Producer Counter* – A counter maintained by hardware and is incremented for every CQE that is written to the CQ. The least significant bits (as indicated by the CQ.*log_cq_*

1. If CPU cache size is larger than CQE-size, for best performance CQ buffer should be aligned on CPU cache line size and size of CQ buffer should be at least one CPU cache line.

size field) of the Producer Counter are referred to as *Producer Index*. The Producer Index points to the next entry to be written by hardware in the CQ. See “CQ Counter” on page 434.

- *Consumer Counter* – A counter maintained by the application software and is incremented for each CQE polled successfully from the CQ. The least significant bits (as indicated by the `CQ.log_cq_size` field) of the Consumer Counter are referred to as *Consumer Index*. The Consumer Index points to the next CQE to be polled. See “CQ Counter” on page 434.

Figure 30: CQ Counter



Both counters are treated as signed integer numbers by SW.

If the *Producer Counter* equals the *Consumer Counter*, then the CQ is empty.

If the *Producer Counter* minus the *Consumer Counter* equals the CQ size, then the CQ is full.

Upon polling the CQ successfully (that is, a new CQE is found), software updates the *Consumer Counter* which is used by hardware to monitor a potential CQ overflow. If CQ is full while another CQE needs to be posted to the CQ, then hardware does the following if overflow detection is enabled:

1. Transfers the CQ to the appropriate error state.
2. Transfers the QP/RQ/SQ to the error state.
3. Generates the appropriate event.

If CQ is full while another CQE needs to be posted to the CQ, and if overflow detection is disabled, then old CQEs may be overwritten.

8.19.1.1 CQE Format

New CQE format for 64B CQE is shown in Table 145.

Table 145 - 64B Completion Queue Entry Format Layout

Offset		0h		4h		8h		Ch		10h		14h		18h		1Ch	
3	1	eth_wqe_id															
3	0	lro_tcp_win															
2	9	lro_min_ttl															
2	8	lro_ack_seq_num															
2	7	rx_hash_result															
2	6	rx_hash_type															
2	5	ml_path															
2	4	smac[47:16]															
2	3	checksum/ib_stride_index															
2	2	sl_roce_pk-type															
2	1	rqp															
2	0	ip_ext_opt															
1	9	cv															
1	8	up															
1	7	ch															
1	6	vid															
1	5	eth_wqe_id															
1	4	lro_tcp_win															
1	3	lro_min_ttl															
1	2	lro_ack_seq_num															
11		rx_hash_result															
1	0	rx_hash_type															
9		ml_path															
8		smac[47:16]															
7		checksum/ib_stride_index															
6		sl_roce_pk-type															
5		rqp															
4		ip_ext_opt															
3		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															
2		smac[47:16]															
2		checksum/ib_stride_index															
2		sl_roce_pk-type															
2		rqp															
2		ip_ext_opt															
2		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															
2		smac[47:16]															
2		checksum/ib_stride_index															
2		sl_roce_pk-type															
2		rqp															
2		ip_ext_opt															
2		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															
2		smac[47:16]															
2		checksum/ib_stride_index															
2		sl_roce_pk-type															
2		rqp															
2		ip_ext_opt															
2		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															
2		smac[47:16]															
2		checksum/ib_stride_index															
2		sl_roce_pk-type															
2		rqp															
2		ip_ext_opt															
2		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															
2		smac[47:16]															
2		checksum/ib_stride_index															
2		sl_roce_pk-type															
2		rqp															
2		ip_ext_opt															
2		cv															
2		up															
2		ch															
2		vid															
2		eth_wqe_id															
2		lro_tcp_win															
2		lro_min_ttl															
2		lro_ack_seq_num															
2		rx_hash_result															
2		rx_hash_type															
2		ml_path															

Table 145 - 64B Completion Queue Entry Format Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											20h				
lro_num_seg												srqn/user_index																								20h
immediate / invalidate key / pkey/flow_table_metadata																																24h				
application												application_opcode										application_info										28h				
byte_cnt/mini_cqe_num																																2Ch				
lro_timestamp_value/timestamp_h																																30h				
lro_timestamp_echo/timestamp_l																																34h				
send_wqe_opcode / rx - drop_counter												qp/dctn/flow_tag																				38h				
wqe_counter																signature										opcode				cqe - format		owner	sc	3Ch		

of extended CQE segment for receive completion report is shown on Table 146.

Table 146 - 128B Completion Queue Entry Format Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											0h-3Ch
GRH / inline_64																												40h-7Ch				
CQE 64B																																

When GRH is placed in the first segment of the extended CQE, it is placed in bytes 0x0 - 0x27.

CQE format is extended in the device to enable to carry operation timestamp as well as atomic response and/or head of the data received on receive operation.

The device CQE format is shown in Table 145.

Each CQ can be independently configured to report either 64 or 128 bytes CQEs by programming *cqe_sz* parameter at CQ creation.

SQ, reporting completions to extended CQ, can be configured to place data of atomic or read response to CQE (instead of writing it to WQE-specified memory buffer) for operations where response fits *Data* section of the CQE (max 64 bytes, which translates to atomic/read of 64 bytes). Note that WQEs on QPs connected to such CQ must request for CQE generation (CE field should be 0x2 or 0x3). See Table 42, “General - Ctrl Segment Format,” on page 312 for details.

The SQ can be configured to place the data always in the inline_64 segment or, if the data size is less than or equal to 32B, place it in the inline_32 segment. Placing data in the inline_64 segment is possible only for CQ configured for 128B CQEs.

RQ, can be configured to place a message to *data* section of CQE, which can be handy to save cache misses on the CPU and chipset. Hardware will only place data to CQE if entire payload fits in the *data* section of CQE. For UD and DC, If GRH exists, it may be scattered into the *data_64* segment of the CQE. If GRH is scattered to CQE, data will not be scattered to the CQE, Data placement to CQE is indicated by *cqe_format and l3_hdr* bits in basic CQE section.

The RQ can be configured to place the data always in the *inline_64* segment or, for RC/XRC QPs only, if the data size is less than or equal to 32B, place it in the *inline_32* segment. Placing data in the *inline_64* segment is possible only for CQ configured for 128B CQEs.

Base section of CQ Entry (CQE) contains enough information to associate the CQE with its WQE and carries enough information for SW to figure out what happened.

The 64B CQE layout is shown in Table 145, “64B Completion Queue Entry Format Layout,” on page 435. The format of basic CQE reporting an Error is shown in Table 159, “Completion with Error CQE Layout,” on page 455. The format of SIGERR_CQE reporting a signature Error is shown in Table 161, “SIGERR CQE Entry Format Layout,” on page 458.

CQE field descriptions are shown in Table 147.

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
2	tls_offload	0x0: NOT_DECRYPTED - the entire pkt has not been decrypted 0x1: DECRYPTED - the entire pkt has been decrypted 0x2: RESYNC - the entire pkt has not been decrypted and hw found magic 0x3: ERROR - found auth err or header err while decrypting			
1	outer_ip_ext_opts	Outer IPv4 Options or IPv6 Extensions presence indication. For IPv4: 1 - IPv4 Options are present in the frame 0 - no IPv4 Options present in the frame. For IPv6: 1 - IPv6 Extensions are present in the frame 0 - no IPv6 Extensions present in the frame Valid only when tunneled bit is set	n	n	y
1	outer_l3_type	Outer packet L3 type: 0x0: IPv4 0x1: IPv6 Valid only when tunneled bit is set	n	n	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IB/EoIB
1	tunneled	Packet is tunneled and successfully passed checksum validations for the outer headers (outer L3 and if available outer L4 checksums). Valid for hypervisor tunneling offloads. SW should use the tunneled field to parse the CQE information correctly, and set TIR.tunneled_offload_en, even if other tunneled packet offloads are not explicitly required or supported.	n	n	y
1	nvmeotcp_resync	See Resync flow in Section 4.10, “NVMeoTCP Offload,” on page 220 . When set, also resync_sn field in the 128B CQE is valid.		n	
1	nvmeotcp_data_digest	Data digest offload passes for all PDUs in this packet.		n	
1	nvmeotcp_zc	Data in this packet has been placed directly in the NVME buffer. This enables the NVMeoTCP fields of the 128B CQE.		n	
16	eth_wqe_id	WQE ID. Valid only for Ethernet traffic Striding WQ.			
1	lro_timestamp_is_valid	This bit is set when CQE’s timestamp field holds LRO timestamp instead of regular timestamp CQE field) – for details please see lro_timestamp_value /timestamp_h and lro_timestamp_echo /timestamp_l CQE’s fields. Valid only for LRO packets (lro_num_seg field not clear)			
1	lro_tcp_psh	LRO: one of the packets in the session had a TCP PUSH flag set Valid only for LRO packets (lro_num_seg field not clear)	n	n	y
2	lro_abrt	LRO abort coalescing reason 0x0:Max_IP_Payload_Size_Exceeded 0x1: LRO_Timeout 0x2: LRO_Exception 0x3: Other Valid only for LRO packets (lro_num_seg field not clear)	n	n	y
4	lro_dup_ack_cnt	Duplicate acknowledgment counter Number of duplicate acknowledgment that were encountered while creating the coalesced segment. Valid only for LRO packets (lro_num_seg field not clear) This field should be always zero.	n	n	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
8	lro_min_ttl	The minimum HopLimit / TimeToLive of all packets in this segment Valid only for LRO packets (lro_num_seg field not clear)	n	n	y
16	lro_tcp_win	TCP window of the last packet in the coalesced segment Valid only for LRO packets (lro_num_seg field not clear)	n	n	y
32	lro_ack_seq_num	LRO acknowledgment sequence number (last packet in session) Valid only for LRO packets (lro_num_seg field not clear)	n	n	y
32	rx_hash_result	RX hash function result Valid only when rx_hash_type is not zero. When tunneled bit is set rx_hash_result can be result of hash on inner or on outer tunneling packet fields. This depends on tunneling type and TIR, which processed the packet, configuration.	n	n	y
8	rx_hash_type	RX hashing performed (this field indicates whether CQE reports all other RX fields. When 0, RX hashing is not valid): rss_hash_type[1:0] - IP source selected • 00 - none • 01 - IPv4 • 10 - IPv6 • 11 - Reserved rss_hash_type[3:2] - IP destination selected • 00 - none • 01 - IPv4 • 10 - IPv6 • 11 - Reserved rss_hash_type[5:4] - L4 source selected • 00 - none • 01 - TCP • 10 - UDP • 11 - IPSEC.SPI rss_hash_type[7:6] - L4 destination selected • 00 - none • 01 - TCP • 10 - UDP • 11 - IPSEC.SPI	n	n	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
1	CE	Congestion Experienced – the packet was marked as congested by the tx overflow mechanism. This field will be set only when tx_lossy_overflow_oper was set to mark. See TBD		n	n
7	ml_path	For responder UD over IB CQE: These are the lower LMC bits of the DLID in an incoming UD packet, higher bits of this field, that are not part of the LMC bits are zeroed by HW. Invalid if incoming message DLID is the permissive LID or incoming message is multicast. Otherwise, this field is reserved.	n	IB only	IB only
32	smac[47:16]	Bits 47:16 of source MAC address of the message. Valid only for RoCE UD and RoCE DCT.	n	RoCE only	n
16	checksum/ ib_stride_index	For ETH: This field is summarized in Section 4.3.1, “Checksum Offload,” on page 189. For IB striding WQ: receive WQE stride index (pointer to start of packet)	n	y	y
16	slid /smac[15:0]	For IB: Source LID of the message. Valid only for UD and DCT QPs. For RoCE: Source MAC address of the message (15-0 bits). Valid only for UD QPs and DCT QPs.	n	y	IB only
1	fl	Force loopback indication [TBD] need to validate this is valid for IPoIB/EoIB	n	IB Only	IB only
2	l3_hdr	L3 Header Presence indication. Valid for responder UD/DC only: 00 - No GRH/IP in the packet 01 - GRH/IP in the packet, located in the receive buffer 10 - GRH/IP in the packet, located in the CQE (in the data_64 / GRH segment) For IB, the L3 header is a GRH always. For RoCE, the L3 header type is indicated in sl_roce_pktype field.	n	y	n

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
4	sl/roce_pktype	For responder UD, DC over IB - the Service Level of the packet. - For responder UD,DC over RoCE:Bit4: UDP -when set R-RoCE frame contains an UDP header otherwise not. - Bits[2:0]:L3_Header_Type. Header type encoding (RoCE) 0x0 :GRH - (RoCE v1.0) 0x1 : IPv6 - (RoCE v1.5/v2.0) 0x2 : IPv4 - (RoCE v1.5/v2.0) Otherwise, this field is reserved.	n	y	IB Only
24	rqpn	For Responder UD QPs, Remote (source) QP number For DC: If CQ.initiator_src_dct==0 ==>DCI number. If CQ.initiator_src_dct==1 ==>Initiator source DCT number.	n	y	IB only
2	hds_type	Header Data Split Type 00 - no split 01 - split at IP Header (ULP Header) 10 - split at TCP/UDP Header (ULP Payload) 11 - split at Look Ahead When tunneled bit is set indicates split for outer packet.	n	n	y
1	l4_ok	If this is bit set that means that HW successfully parsed TCP/UDP packet and that packet's L4 checksum was successfully validated. otherwise this bit is disabled. When tunneled bit is set indicates checksum validation status of the inner L4 header.	n	n	y
1	l3_ok	If this bit is set that means that HW successfully parsed IP packet, the packet successfully passed HW checks and that packet's IP checksum was successfully validated, otherwise this field is disabled. When tunneled bit is set indicates checksum validation status of the inner IP header.	n	n	y
1	l2_ok	L2 known headers are identified according to a set of HW checks. When tunneled bit is set refers to inner frame. TBD: verify this with HW	n	n	y
1	ip_frag	The bit is set if packet is fragmented (indication is set for IPv4/IPv6 accordingly). When tunneled bit is set, this bit refers to inner packet.	n	n	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SO	RQ IB/RoCE	RQ Raw Eth/IPv6/EoIB
3	l4_hdr_type	0x0: None 0x1: TCP - TCP header was present in the packet 0x2: UDP - UDP header was present in the packet 0x3: TCP_EMPTY_ACK - TCP header was present in the packet with Empty TCP ACK indication. (TCP packet <ACK> flag is set, and packet carries no data) 0x4: TCP_WITH_ACL - TCP header was present in the packet with TCP ACK indication. (TCP packet <ACK> flag is set, and packet carries data). When tunneled bit is set, this field describes the inner packet.	n	n	y
2	l3_hdr_type	L3 Header Type 00 - None 01 - IPv6 10 - IPv4 When tunneled bit is set, this field describes the inner packet.	n	n	y
1	ip_ext_opts	For IPv4: 1 - IPv4 Options are present in the frame 0 - no IPv4 Options present in the frame For IPv6: 1 - IPv6 Extensions are present in the frame 0 - no IPv6 Extensions present in the frame For non IP packet (l3_hdr_type == none) undefined. Supported Ipv6 extensions in the device: Hop By Hop, Routing, Fragment, Destination, Authentication, Mobile When tunneled bit is set, this field describes the inner packet.	n	n	y
1	cv	Customer VLAN (cVLAN) Valid for RoCE UD/DC and networking service: EoIB, Raw Eth When set - cVLAN was present in the incoming frame and is reported by the CQE fields: up, cfi, vid When tunneled bit is set, the cv refers to the outer Eth header	n	RoCE Only	EoIB/Raw Eth only
3	up	802.3 priority field. Valid when cv is set. Valid for RoCE UD/DC and networking service: EoIB, Raw Eth When tunneled bit is set, the cv refers to the outer Eth header	n	RoCE Only	EoIB/Raw Eth only

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
1	cfi	802.3 CFI field. Valid when cv is set. Valid for RoCE UD/DC and networking service: EoIB, Raw Eth When tunneled bit is set, the cv refers to the outer Eth header	n	RoCE Only	EoIB/Raw Eth only
12	vid	802.3 VLAN-ID field. Valid when cv is set. Valid for RoCE UD/DC and networking service: EoIB, Raw Eth When tunneled bit is set, the cv refers to the outer Eth header	n	RoCE Only	EoIB/Raw Eth only
8	lro_num_seg	Number of coalesced segments of LRO. When clear - LRO was not performed.	n	n	y
24	srqn/user_index	If HCA_CAP.cqe_version==1: The user_index which software sets when creating a context (RQ/SQ/QP/DCT/XRC_SRQ). If HCA_CAP.cqe_version=0: Indicates the SRQ number for responder. SRQN==0 - message was not received on SRQ	y	y	y
32	immediate / invalidate key / pkey/ flow_table_meta- data	For a responder CQE, if completed WQE Opcode is Send With Immediate or Write With Immediate, this field contains immediate field of the received message. For a responder CQE, if completed WQE Opcode is Send With Invalidate, this field contains the R_key that was invalidated. For a responder CQE of a GSI packet, bits 15:0 of this field contains the Pkey of the packet. For Eth RQ, this field holds metadata collected during packet flow table process in metadata_reg_b.	n	y	y
8	application	Application specific cqe. 0x1: tag_matching - valid only when QP is tag_matching qp		y	

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
8	application_opcode	Application opcode. When application==tag_matching 0x1: tag_consumed - tag was consumed. 0x2: message_arrived_expected - expected message arrived data is in the matching entry buffer. 0x3: message_arrived_unexpected - unexpected message arrived to UWQ. XRQ.hw_phase_cnt was advanced by 1. 0x4: message_arrived_no_tag- non tagged message arrive to UWQ. 0x5: Append operation 0x6: Remove operation 0x7: NOP operation 0x9: tag_consumed_sw_rndv - tag was consumed. HW did not perform rndv completion. 0xA: tag_consumed_message_arrive - Tag was consumed and expected message arrived. Data is in the matching entry buffer. 0xB: tag_consumed_message_arrive_sw_rndv - Tag was consumed and expected message arrived. HW did not perform rndv completion. 0xC: message_completion_canceled – message is in matching entry buffer. Device cannot complete the offload.	y	y	
16	application_info	Application extended information. For application ==tag_matching When application_opcode==tag_consumed or tag_consumed_sw_rndv this field holds the tag_index that was consumed.			

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
32	byte_cnt/mini_cqe_num	Byte count of data transferred. Applicable for RDMA-read, Atomic and receive completions. Byte_cnt specifies total byte count scattered to the buffer or the CQE (if scatter-to-CQE is enabled). For RoCE/IB: If GRH/IPv4/IPv6 is not scattered to the CQE, its 40B length is included in the byte count (even if GRH is not present). For Header Data Split: Receive Queue that is subject for headers-separation byte_cnt[31:24] specifies the number of bytes scattered to the first scatter entry (headers' length). byte_cnt[23:0] specifies the total byte count received (without headers). For LRO messages (lro_num_seg field not clear): the total length of messages is specified. For compressed CQE (cqe_format == 3) the total number of Mini CQEs is specified (will be > 1). For Striding WQ: See Section 8.9.1.3, "Striding Receive WQ," on page 304. byte_cnt[31:31] == 1 indicates the CQE is of type FILLER CQE. byte_cnt[30:30]: specify end of message (for IB Only). byte_cnt[29:16]: specify number of consumed strides. byte_cnt[15:0]: specify bytes count received including padding. Not valid for FILLER CQE and ERROR CQE. For padded messages, the padding is excluded.	y	y	y
32	lro_timestamp_value/ timestamp_h	When lro_timestamp_is_valid is on: This field holds the Timestamp Value (TSval) field of the header of the last coalesced segment. When lro_timestamp_is_valid is off: This field holds the MSB of 64-bit sample of the internal timer taken when this CQE was generated. See Section 8.19.12, "CQE Timestamping," on page 471	y	y	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
32	lro_time-stamp_echo/ timestamp_l	When lro_timestamp_is_valid is on: This field holds the Timestamp Echo Reply (TSecr) field of the header of the latest coalesced segment with valid TSecr field. When lro_timestamp_is_valid is off: This field holds the LSB of 64-bit sample of the internal timer taken when this CQE was generated. See Section 8.19.12, “CQE Timestamping,” on page 471	y	y	y
8	rx_drop_counter	On UD/UC/Raw Eth responder - the number of dropped packets because of no RCV WQE since the last CQE [TBD] check if in Golan it didn't count UC, if this is the case, then need to add capability bit	n	y	y
8	send_wqe_op-code	On requester - the send WQE opcode	y	n	n
24	QPN/DCTN/ flow_tag	If HCA_CAP.cqe_version==0: • QPN/DCTN If HCA_CAP.cqe_version==1: • For IB, Indicates QPN/DCTN. • For Raw Ethernet, Indicates the flow_tag which software sets when creating a flow_entry. For IPoIB: If flow_tag was set in one of the flow tables, the field indicates the flow_tag (provided by the driver on flow entry creation) of the last table with a valid flow_tag field.	n	y	y
16	wqe_counter	wqe_counter of the WQE completed. • For SQ: the sq_wqe_counter is reported (“Send Queue” on page 300) • For RQ: the rq_wqe_counter is reported (“Receive Queue” on page 302) • For SRQ: the wqe_index is reported (“Shared Receive Queue” on page 306) For Ethernet traffic Striding WQ: receive WQE stride index (pointer to start of the message)	y	y	y
8	Signature	Byte-wise XOR of CQE - signature protection (see “Completion and Event Queue Elements (CQEs and EQEs)” on page 879). CQE is valid if byte-wise XOR of entire CQE (including <i>signature</i> field) and the CQE index is 0xff0xFF. For 128B CQE, the GRH/inline_64 section is taken into account if data / GRH was written to it (cqe_format == 2 or grh == 2)	y	y	y

Table 147 - Completion Queue Entry Fields

Bits	Name	Description	SQ	RQ IB/RoCE	RQ Raw Eth/IPoIB/EoIB
4	Opcode	0x0: Requester 0x1: Responder_RDMA_Write_with_Immediate 0x2: Responder_Send 0x3: Responder_Send_with_Immediate 0x4: Responder_Send_with_Invalid 0x5: Resize_cq - See “Resizing a CQ” on page 453 0x6: No_packet_operation 0x12: SIGERR- Signature Error CQE,(SIGERR_CQE format shown in Table 161, “SIGERR CQE Entry Format Layout,” on page 458). 0xD: Requester_Error - (ERR_CQE format shown in Table 8.19.7, “Completion With Error,” on page 455) 0xE: Responder_Error -(ERR_CQE format shown in Table 8.19.7, “Completion With Error,” on page 455) 0xF: Invalid_CQE - Ownership bit is not valid and the CQE is in HW ownership.	y	y	y
2	cqe_format	0 - no inline data in the CQE 1 - inline data in the data_32 segment of the CQE (offsets 00h-1Ch) 2 - inline data in the data_64 segment of the CQE (additional 64B) 3 - Compressed CQE Please note that if cqe_format == 3, mini_cqe_num > 1	y	y	y
1	SE	Solicited event. This CQE cause EQE generation for solicited event	y	y	y
1	Owner	CQE ownership bit. See Section 8.19.3.1, “CQE Ownership,” on page 450.	y	y	y

8.19.1.2 NVMETCP Extra 64Bytes CQE

NVMEoTCP includes an extra 64 Bytes of information in addition to the basic 64B CQE.

Table 148 - NVMeoTCP Extra CQE Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
zero_copy_region_size																zero_copy_region_start_offset																00h
cmd_data_offset_or_resync_tcp_sn																																04h
ccid																																08h
																																0Ch-3Ch

Table 149 - NVMeoTCP Extra CQE Info Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	zero_copy_region_size	Number of Bytes from the packet were written to the NVME buffer directly. Valid only when CQE.nvmeotcp_zc==1	
	15:0	zero_copy_region_start_offset	The offset in the packet where zero copy began. Valid only when CQE.nvmeotcp_zc==1	
04h	31:0	cmd_data_offset_or_resync_tcp_sn	If CQE.nvmeotcp_zc==1, then this field is the offset in the NVME buffer where data has been placed. CQE.nvmeotcp_resync==1, the this is the TCP sequence number where the NVME header has been speculatively found.	
08h	31:16	ccid	The CCID of the PDU whose data is placed directly into the NVME buffer	

8.19.1.3 CQE Format For Application Offload

For application offload (like tag matching application offload), the first 16 bytes of the 64Byte CQE report a specific “application header” depending on the application_opcode field.

8.19.1.3.1 Tag Matching Application Header

For CQE of Tag Matching offload (application==tag_matching), the reported application headers are:

1. For tag_consumed application_opcodes:

Table 150 - Tag Consumed Application Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
TMH																																00h-0Ch

Table 151 - Tag Consumed Application Header Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	TMH	Tag matching header	

2. For Append/Remove/Nop application_opcodes:

Table 152 - List Operation Application Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
success																															00h	
	hw_phase_cnt																														004h	
																															08h-0Ch	

Table 153 - List Operation Application Header Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	success	Valid only for remove operation. If set then remove success.	
04h	31:16	hw_phase_cnt	<i>hw_phase_cnt</i> value at the end of the operation execution in HW. Valid only for remove, append and NOP operations	

8.19.2 CQ DoorBell Record

CQ Doorbell Records are located in physical address pointed to by the CQ Context. CQ Doorbell Records are aligned on an 8B boundary. Software updates the *update_ci* after CQEs are reaped from the CQ (Poll for Completion verb). Software updates *cmd_sn*, *cmd* and *cq_ci* when CQ is armed for event generation (Request Completion Notification verb).

CQ Doorbell record layout is shown in Table 154.

Table 154 - CQ Doorbell Record Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								update_ci																							0h	
			cmd_sn					arm_ci																							4h	

Table 155 - CQ DoorBell Record Field Descriptions

Offset	Bits	Name	Description	Access
0h	23:0	update_ci	Consumer counter of the last polled CQE	
04h	29:28	cmd_sn	Command Sequence Number. The value of this field should be initialized by 2 when created. This value should be '0 for the first DoorBell rung, and should be increment on each first DoorBell rung after a completion event. That is, cmd_sn = num_of_completion_event_delivered% 4	
	24	cmd	1 - Request notification for next Solicited completion event. arm_ci field should specify the CQ Consumer Counter at the time of the DoorBell ring. 0 - Request notification for next Solicited or Unsolicited completion event. Counter field should specify the CQ Consumer Counter at the time of the DoorBell ring.	
	23:0	arm_ci	Consumer Counter for arming CQ as described in <i>cmd</i> .	

8.19.3 Poll for Completion

8.19.3.1 CQE Ownership

CQE ownership is defined by **Owner** bit in the CQE. The value indicating SW ownership is flipped every time CQ wraps around. The following pseudo code explains how to determine that a CQE is in SW ownership:

```
ownership cqe_ownership(cqe) {
if (cqe.owner == ((consumer_value >> log2_cq_size) & 1)) {
return SW ownership
}
else {
return HW ownership
}
}
```

}

When a CQ is created or resized, software needs to initialize the entire CQE buffer with owner bit = Hardware.

8.19.3.2 Reporting Completions

While executing a poll for completion, software examines the **Owner** bit and checks that the opcode field is different than 0xf. of the CQE entry pointed by *consumer_counter*. If the **Owner** entry value is SW, the CQE is valid, and software should consume the entry and increment *Consumer Counter* in the CQ Doorbell Record. This operation must be repeated as long as the **Owner** bit of the CQEs examined has the SW value. Pseudo-code for the Poll for Completion algorithm is shown below:

```
If (cqe_ownership(CQE[Consumer_idx]) == SW) & (CQE,opcode != 0xF) {
    new_CQE = consume_entry(); // Read entry from the queue
    consumer_counter++; // Increment consumer counter
    update_DoorBell_record(CQ,consumer_counter); // Update DoorBell record
    return new_CQE;
} else {
    return CQ_EMPTY;
}
```

Note: After seeing that the CQE is in SW ownership, SW should do memory barrier and re-read the CQE.

Note: For multiple CQE polling, it is allowed to update the DoorBell record only once when exiting the algorithm.

8.19.4 Request Completion Notification

Subscription for this event is achieved by updating a DoorBell record and writing the appropriate command to the CQ Doorbell Register (“UAR Page Format” on page 250). Users can request events for Solicited event notification or Unsolicited event notification.

In this section:

- “Request notification for next completion” will be denoted by *ARM_NEXT*
- “Request notification for next Solicited completion” will be denoted by *ARM_SOLICITED*

The CQ can be armed by application software to generate an event upon completion of a Work Request on the WQ mapped to the CQ. The Completion Event is generated by the CQ if a new CQE arrives at the CQ and *one* of the following conditions is met:

1. *ARM_NEXT* was executed on the CQ and completion has been generated on this CQ
2. *ARM_SOLICITED* was executed on the CQ and a WQ mapped to this CQ has either received an IB message with the **Solicited Event** bit set, or Send WQE which completed with the **E** bit set

Completion events are reported to an EQ by writing an EQE, and no further events are generated before a new request for notification with a higher value of *cmd_sn* is executed. The required usage of *cmd_sn* is described in “CQ DoorBell Record” on page 449.

Whenever a Completion Notification is requested by the verbs consumer, software should decide if it is going to ring the DoorBell. If the last rung *cmd_sn* is different than the *cmd_sn* to be rung, then a DoorBell should be rung. Otherwise, if the two *cmd_sn* values are equal, then Table 156 applies.

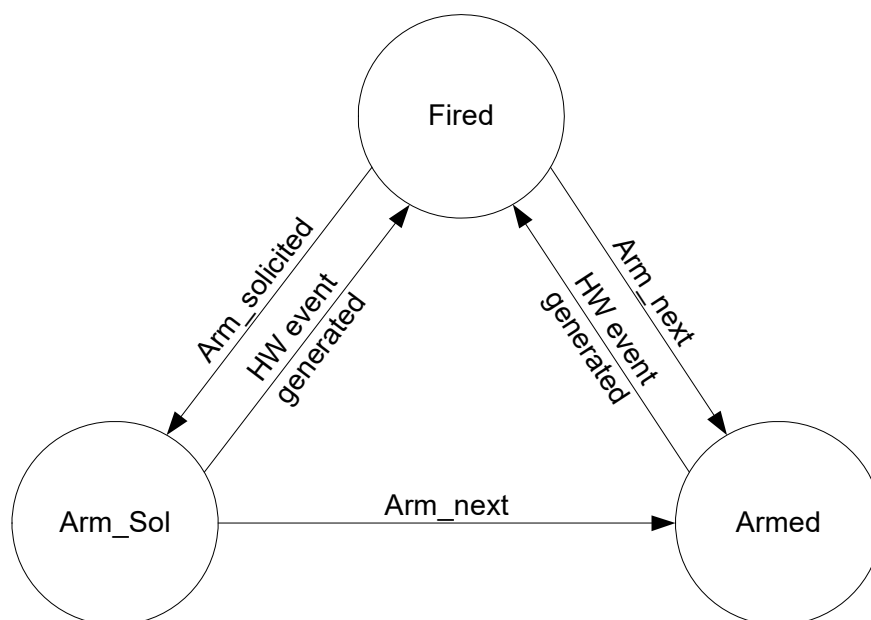
Table 156 - ARM CQ DoorBell Ringing With Repeated *cmd_sn*

		<u>Command of Last Executed ARM</u>	
		ARM_NEXT	ARM_SOLICITED
<u>Command Requested</u>	ARM_NEXT	ring	ring
	ARM_SOLICITED	do not ring	ring

If new CQEs are posted to the CQ after the reporting of a completion event and these CQEs are not yet consumed, then an event will be generated immediately after the request for notification is executed.

Event generation and handling is controlled by the Event State machine. Its transitions are shown in Figure 31.

Figure 31: CQ Event Delivery State Machine



Mellanox HCAs keep track of the last index for which the user received an event. Using this index, it is guaranteed that an event is generated immediately when a request completion notification is performed and a CQE has already been reported.

In addition, HCA provides mechanism to moderate events generated by the CQ as detailed in “Completion Event Moderation” on page 508.

8.19.5 Completion Queue Update Error

CQ updates can cause errors resulting from software bugs. In particular, a CQ overrun is checked while posting a completion, and if encountered, the QP/RQ/SQ is transferred to the appropriate error state. Hardware will deliver the auxiliary information to the software in an EQE written to the EQ.

CQ update (and error discovery) are not synchronized with WQE execution. Thus, a positive acknowledgment can be sent to the remote request, and subsequently a CQ overrun (or other error) may be detected. Hardware may not write the CQE as a result of the error, and the CQ will be transitioned to an error state as well as the QP/RQ/SQ that caused the CQ error. To avoid these errors, software must ensure that the number of WQEs posted to a QP/RQ/SQ does not exceed the number of entries in the CQ associated with this QP/RQ/SQ. When multiple QPs/RQs/SQs post completions into the same CQ, this check must be cumulative.

8.19.6 Resizing a CQ

This feature enables modifying CQ size or CQEs size, driver must not modify any other field in CQC when performing this flow.

In order to resize the CQ or CQEs size, SW needs to allocate a new buffer for the CQ and pin it. Then, by submitting a MODIFY_CQ command with op_mod =1 (Resize CQ).

In this command SW needs to:

1. Pass the physical pages - as it was done while opening the CQ.
2. Fill below fields in the CQ context:
 - a. log_cq_size
 - b. page_offset.
 - c. log_page_size

Resizing Cq is enabled just if resize_cq_en =1 in QUERY_HCA_CAP command (see Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667).

While processing the command, the hardware will report a special CQE called Resize_Cq (CQE.opcode = 5h) to the old CQ denoting that the resizing operation has been completed. Note that this can be observed before or after the MODIFY_CQ command completes. Therefore, it is advised that the driver reserves room for one extra CQE when creating a resizable CQ.

Once the special CQE has been observed, driver can unpin the old CQ buffer (regardless whether this happens before or after the MODIFY_CQ command completes).

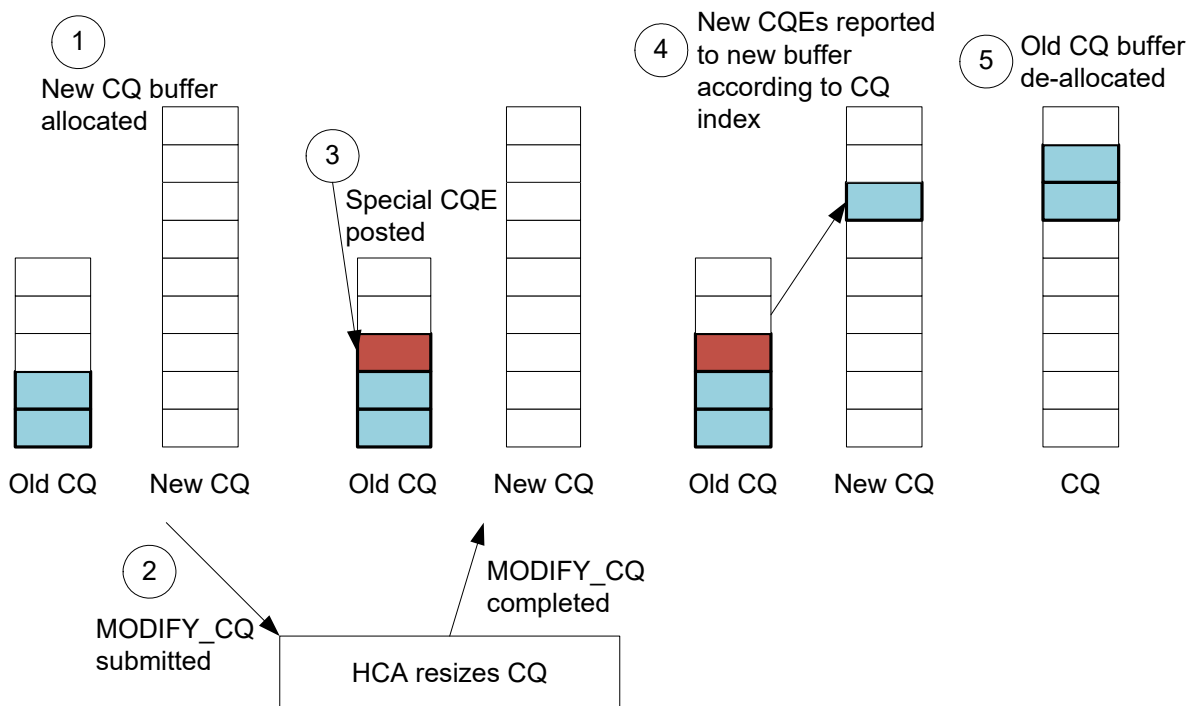
When the MODIFY_CQ command completes, HW will not attempt to access the old buffer, Note that it can still contain valid CQEs; therefore, SW should still poll this buffer for CQEs until reaching the special CQE that indicates to move to the new buffer.

Note that the `MODIFY_CQ` command, used for resizing the CQ, cannot be nested; thus, it is not possible to issue a new `MODIFY_CQ` command, with the `Resize CQ` opcode modifier, to the same CQ before the previous one is completed.

When moving to the new CQE buffer, the index where SW should start polling at is calculated based on the consumer counter and the new CQ size. Thus, the software needs to take the $\log_2 c\_q\_size$ least significant bits of the `consumer_counter` and use them to reference the first CQE in the new buffer. The ownership bit is calculated similarly.

Figure 32 illustrates the resizing process.

Figure 32: CQ Resize



It is recommended that when the `MODIFY_CQ` command ends, or before starting any new resize operation, to copy the CQEs from the old buffer to the new buffer. The process involves going through the old buffer, look for the special CQE, and then copying all prior CQEs to the new CQ omitting the special CQE. This simplifies accounting issues with the amount of CQEs that are produced by this CQ (total of at most one extra CQE, the resize special CQE, at all times).

The process of arming the CQ is unaffected by the resizing operation since the consumer/producer counters remain intact.

The format of the special resizing CQE is shown in Table 157.

Table 157 - Resize CQE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-2Ch

Table 157 - Resize CQE Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
timestamp																																30h-34h	
																																38h	
																signature								opcode								owner	3Ch

Table 158 - Resize CQE Field Descriptions

Offset	Bits	Name	Description
30h-34h	64	timestamp	
3Ch	15:8	signature	
	7:4	opcode	
	0	owner	

TBD - add description table

8.19.7 Completion With Error

Completions with error are reported to the CQ similarly to regular completions. In the case of completion with error, not all the CQE fields are valid and some of the fields have different meanings. For every WQE that completes in error, a completion with error is generated; “CQE with error” is specified by 13 or 14 value in the *opcode* field.

The layout of a CQE with error entry is shown in Table 159.

Table 159 - Completion with Error CQE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																eth_wqe_id																00
																																04-10h
ib_stride_index																																14h
																																18h-1Chh

Table 159 - Completion with Error CQE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								srqn_or_user_index																								20h
																																24h-28h
byte_cnt																																2Ch
																																30h
hw_error_syndrome								hw_syn-drome_type								vendor_error_syndrome								syndrome								34h
S_WQE_OPCODE								qpn_or_dctn_or_flow_tag																								38h
wqe_counter																signature								OPCODE						owner	3Ch	

Table 160 - Completion with Error CQE Field Description

Offset	Bits	Name	Description	Access
00h	15:0	eth_wqe_id	Same description of the same field can be found in Table 147, "Completion Queue Entry Fields," on page 437.	
14h	31:16	ib_stride_index	Same description of the same field can be found in Table 147, "Completion Queue Entry Fields," on page 437.	
20h	23:0	srqn_or_user_index	Same description of the same field can be found in Table 147, "Completion Queue Entry Fields," on page 437.	
2Ch	31:0	byte_cnt	Same description of the same field can be found in Table 147, "Completion Queue Entry Fields," on page 437.	
34h	31:24	hw_error_syndrome	Mellanox specific error HW syndrome. See separate documentation.	
	23:20	hw_syndrome_type		

Table 160 - Completion with Error CQE Field Description

Offset	Bits	Name	Description	Access
34h	15:8	vendor_error_syndrome	Mellanox specific error syndrome. Undocumented. For further information contact Mellanox technical support.	
	7:0	syndrome	IB compliant completion with error syndrome 0x1: Local_Length_Error 0x2: Local_QP_Operation_Error 0x4: Local_Protection_Error 0x5: Work_Request_Flushed_Error 0x6: Memory_Window_Bind_Error 0x10: Bad_Response_Error 0x11: Local_Access_Error 0x12: Remote_Invalid_Request_Error 0x13: Remote_Access_Error 0x14: Remote_Operation_Error 0x15: Transport_Retry_Counter_Exceeded 0x16: RNR_Retry_Counter_Exceeded 0x22: Aborted_Error other is Reserved Syndrome is defined according to the <i>InfiniBand Architecture Specification, Volume 1</i> . For a detailed explanation of the syndromes, refer to the <i>Software Transport Interface</i> and <i>Software Transport Verbs</i> chapters of the IB specification.	
38h	31:24	S_WQE_OP-CODE	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	
	23:0	qp_n_or_dctn_or_flow_tag	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	
3Ch	31:16	wqe_counter	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	
	15:8	signature	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	
	7:4	OPCODE	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	
	0	owner	Same description of the same field can be found in Table 147, “Completion Queue Entry Fields,” on page 437.	

Once an QP/RQ/SQ has transitioned to the Error state, subsequent WQEs posted to the QP/RQ/SQ will result in reporting a completion with error (flushed) to the associated CQ.

8.19.8 SIGERR CQE - Signature Error CQE

SIGERR CQEs are reported to the CQ in a similar fashion to regular completions. This type of CQE is generated for the first signature error the device discovers per Mkey.

SIGERR CQE is specified by 12 value in the *Opcode* field.

This feature is enabled when *sigerr_cqe* in device capabilities is 1. SIGERR CQE format is shown in Table 161.

Table 161 - SIGERR CQE Entry Format Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	0											0h
																															4h
																															8h
																															Ch
																															10h
																															14h
																															18h
																															1Ch
																															20h
																															24h
																															28h
																															2Ch
																															30h
																															34h
																															38h
																															3Ch

SIGERR CQE fields descriptions are shown in Table 162.

Table 162 - SIGERR CQE Field Descriptions

Offset	Bits	Name	Description
10h	31:0	Expected Transient Signature	For T10-DIF, T10-DIF CSUM See Table 691, “Transient Signature Layout - For T10 - DIF, T10 - DIF CSUM,” on page 899 For XOR32,CRC32 signature See Table 692, “Transient Signature Layout - For XOR32, CRC32 signature,” on page 900 For XOR8 signature see See Table 693, “Transient Signature Layout - For XOR8 signature,” on page 900

Table 162 - SIGERR CQE Field Descriptions

Offset	Bits	Name	Description
14h	31:0	Actual Transient Signature	For T10-DIF, T10-DIF CSUM See Table 691, “Transient Signature Layout - For T10 - DIF, T10 - DIF CSUM,” on page 899 For XOR32,CRC32 signature See Table 692, “Transient Signature Layout - For XOR32, CRC32 signature,” on page 900 For XOR8 signature See Table 693, “Transient Signature Layout - For XOR8 signature,” on page 900
18h	31:0	Expected Reference Tag	The expected reference tag - valid only for T10-DIF
1Ch	31:0	Actual Reference Tag	The received reference tag - valid only for T10-DIF
20h	31:16	Syndrome	Syndrome is a bitmask, in which each bit has the following meaning: 0-11: reserved 11 - reference_tag_error: mismatch was detected in the reference tag for DIF. 12 - application_tag_error: mismatch was detected in the application tag for DIF. 13 - block_guard_error: mismatch was detected in the CRC calculation for DIF or XOR,CSUM,CRC,for all other signature types. 14-15: reserved
24h	31:0	Mkey	Memory key
28h	31:0	Signature Error Byte Offset_H	Signature Error Byte offset[63:32]
2Ch	31:0	Signature Error Byte Offset_L	Signature Error Byte offset[31:0]

8.19.9 CQE Compressing

CQE Compressing reduces PCI overhead by coalescing and compressing multiple CQEs into a single merged CQE. Successful compressing improves message rate especially for small packet traffic.

CQE compressing is supported for all 64B CQE formats (with certain limitations) generated by RQ/Responder or by SQ/Requester.

[TBD - add indication that this feature supported to Networking off-loadsTerminology](#)

8.19.9.1 CQE Compressing

Instead of writing full CQEs to memory, multiple almost identical CQEs are merged and compressed. Information that is shared between the CQEs is written once regardless of the number of compressed CQEs. In addition, only the unique information s(mall amount of bytes compared to full CQE size) is written per CQE.

8.19.9.2 CQE Compression Block

This block contains multiple compressed CQEs. CQE Compression Block contains a single copy of CQEs properties which are shared between all the compressed CQEs (called Title, see below) and multiple mini CQEs (CQEs in compressed form).

8.19.9.3 Title

The Title holds information which is shared between all the compressed CQEs in the CQE Compression Block. In each Compression Block there is only a single Title regardless of the number of compressed CQEs.

8.19.9.4 Mini CQE

A CQE in compressed form that holds some data needed to extract a single full CQE, for example 8 Bytes instead of 64 Bytes.

The shared information between all compressed CQEs, which belong to the same CQE Compression Block called Title, is written once, and only the unique information in each compressed CQE, for example 8 bytes, is written per compressed CQE, called mini CQE.

8.19.9.5 CQE Compressing Configuration

If CQE compressing is supported (`HCA_CAP.cqe_compression == 1`), enabling/disabling CQE compressing can be performed per CQ by setting `CQ.cqe_compressing_en` to 1/0 when the CQ is created. `CQ.mini_cqe_res_format` should be configured in order to decide the format of the mini CQE in the responder side.

Note that CQE compression for 128 Byte cqe is supported only when `HCA_CAP.cqe_compression_128b == 1`.

8.19.9.6 CQE Compression Block

8.19.9.6.1 CQE Compression Block Format

Compression CQE Block consists of the following three components of 64 Bytes each:

- Title:
 - Can be recognized by `CQE.cqe_format == 3`
 - Title CQE looks like an “ordinary” 64 Byte CQE besides:
 - Byte Count Field indicates the total number of Mini CQEs.
 - `cqe_format == 3`
 - Most of the fields which appear in the title are identical in all CQEs which are compressed together.
 - Regardless of the number of mini CQEs each CQE Compression Block contains only a single title
 - Title is located at offset 0 in the compressed CQE.

- Mini CQE Array:
 - Contains 8 mini CQEs - mini CQE Formats are described in Section 8.19.9.7, “Mini CQE Formats,” on page 462.
 - Mini CQE arrays offsets: offset 1 for the first array, and multiple of 8 for the other arrays.
 - Number of mini CQEs arrays is: $(\text{mini\_cqe\_num}/8) + 1$
- Reserved CQEs:
 - HW does not write to these CQEs but only skips them.
 - The number of reserved CQEs (each of 64 Bytes): $\text{mini\_cqe\_num} - \text{number\_of\_mini\_cqe\_arrays} - 1$
 - Reserved CQEs are located in all the remaining locations of the CQE Compression Block which are not occupied by the title and mini CQE arrays.

Title, mini CQEs arrays and reserved CQEs can be optionally used by SW as a location to write the extracted CQEs but this is not obligatory and depends on SW CQE extraction implementation.

Title is scattered at the offset 0 of the CQE Compression Block. After the Title, mini CQEs array is scattered at offset 1. Mini CQEs array always occupies 64 bytes and contains maximum 8 valid mini CQEs. SW can know which entries in an array are valid by using the total number of mini CQEs which is reported in the Title. If the number of mini CQEs is > 2 then reserved CQEs are allocated ($\max(8 - 2, 6)$), HW does not scatter these CQEs. It just skips them when it compresses CQEs. If there are more than 8 mini CQEs, additional mini CQE arrays are written at the offset which starts the next batch of Mini CQEs, i.e. the offsets are: 8, 16, 24... and after each Mini CQE array appears reserved CQEs. After each not first and not last mini CQE array, there is exactly 7 reserved CQEs. After last mini CQE array there are between 0 till 7 reserved CQEs and after first Mini CQE array there are between 0 till 6 reserved CQEs. The exact number of mini CQE arrays is a function of the number of compressed CQEs as described above.

8.19.9.6.2 Mini CQE with Striding RQ

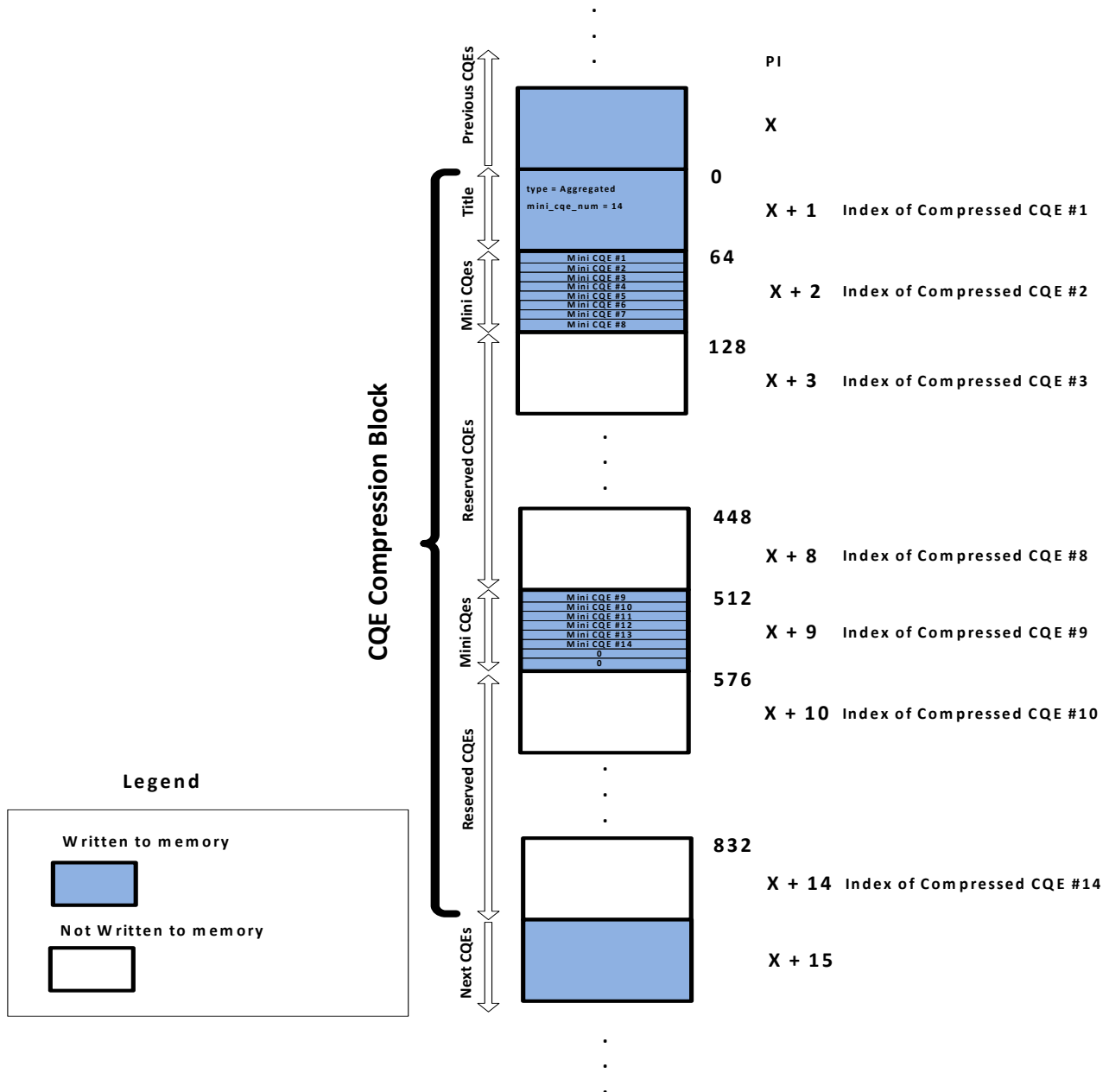
With `CQ.mini_cqe_res_format=0x3`, CQEs of an STRWQ RQ may be compressed only if all the packets that generated them fit into a single STRWQ stride. In this case, the stride index of mini-CQE N (where N=1 denotes the first mini-CQE in the CQE compression Block) is calculated by SW by adding N to the stride index reported in the title CQE (in the title's `CQE.wqe_counter` field).

When `HCA_CAP.mini_cqe_Resp_stride_index=1`, the device supports `CQ.mini_cqe_res_format=0x3`. With `CQ.mini_cqe_res_format=0x3`, each mini-CQE of an STRWQ RQ includes the stride index. In this case, the device may compress CQEs of an STRWQ RQ also if the packets that generate them have different sizes. For mini-CQEs of a legacy RQ associated with a CQ configured with `CQ.mini_cqe_res_format=0x3`, the `stride_index` field is not valid.

8.19.9.6.3 CQE Compression Block Example

Figure 33, “CQE Compression Block Example” provides an example of a CQE Compression Block which contains 14 mini CQEs. In this example, HW successfully coalesced and compressed 14 CQEs.

Figure 33: CQE Compression Block Example



8.19.9.7 Mini CQE Formats

For Responder, there are two formats of mini CQE depending on CQ.mini_cqe_res_format.

Mini Cqe with Rx Hash (when CQ.mini_cqe_res_format==0) and Mini Cqe with Checksum (when CQ.mini_cqe_res_format==1).

Table 163 - Responder Mini CQE With Rx Hash Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_hash_result																															00h	
byte_count																															04h	

Table 164 - Responder Mini CQE With Rx Hash Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	rx_hash_result		
04h	31:0	byte_count		

Table 165 - Responder Mini CQE With Checksum Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
checksum																																00h
byte_count																															04h	

Table 166 - Responder Mini CQE With Checksum Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	checksum		
04h	31:0	byte_count		

Table 167 - Responder Mini CQE With Checksum and Stride Index Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
checksum																stride_index																00h
byte_count																															04h	

Table 168 - Responder Mini CQE With Checksum and Stride Index Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	stride_index		
	31:16	checksum		
04h	31:0	byte_count		

For Requester, mini CQE format described in Table 169, “Requester Mini CQE Layout,” on page 464

Table 169 - Requester Mini CQE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
wqe_counter																s_wqe_opcode												00h				
byte_count																															04h	

Table 170 - Requester Mini CQE Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	wqe_counter		
	15:8	s_wqe_opcode		
04h	31:0	byte_count		

8.19.9.8 Consumer/Producer Counter

HW increments producer counter for each compressed CQE as an ordinary CQE. Note that the Title is written at the index of the first compressed CQE and first mini CQE array at the index of the second compressed CQE. The second mini CQE array is at index of the ninth compressed CQE and so on.

SW must increment consumer counter for each compressed CQE as when it handles as an “ordinary” CQE.

If SW decides to update the DoorBell Record before completing processing all the Mini Cqes of the CQE Compression Block, SW is responsible to take a copy of the Title and last Mini Cqes Array which has a valid unprocessed Mini CQEs, Since HW can overwrite these CQ entries.

8.19.9.9 Compressed CQEs Fields Extraction

If SW detects a CQE Compression Block by `cqe_format == 3`, SW should handle the Mini CQEs in the same order they were compressed.

For each mini CQE, SW gets or calculates the information required as described below:

1. Get the Title:
`base_index = consumer_counter & log2_cq_size`
`Title = CQ[base_index]`
2. Get the number of Mini CQEs:
`Title.mini_cqe_num` holds the total number of Mini CQEs. Please note that when `CQE.inline_scatter == 3` `CQE.mini_cqe_num` fields will be > 1 .
3. Get compressed CQEs Fields:
 Let's assume that SW "decompresses" mini CQE #i:
 For Responder all CQE fields besides `byte_cnt`, `checksum`, `wqe_counter`, `cqe_format` and if `CQ.mini_cqe_res_format == 1` also `rx_hash_result` are taken as is from the title.
 For Requester all CQE fields besides: `byte_cnt`, `type`, `wqe_counter` and `s_wqe_opcode` are taken as is from the title.
 - Get Mini CQE Entry:
`mini_cqe_array_offset = i < 8 ? 1 : 8*(i / 8)`
`mini_cqe_array = CQ[base_index + mini_cqe_array_offset];`
`mini_cqe_entry = mini_cqe_array[i % 8];`
 - Calculate `wqe_counter`:
 For responder: Since `wqe_counters` of mini CQEs are sequential, the WQE counter of the CQE can be calculated in the following way: `wqe_counter = title.wqe_counter + i`.
 For requester: `mini_cqe_entry.wqe_counter`.
 - `s_wqe_opcode`: (for requester only)
`s_wqe_opcode = mini_cqe_entry.s_wqe_opcode`
 - `byte_count`:
`byte_count = mini_cqe_entry.byte_count`
 - `rx_hash_result`: (For responder only)
 If `CQ.mini_cqe_res_format == 0`:
`rx_hash_result = mini_cqe_entry.rx_hash_result;`
 If `CQ.mini_cqe_res_format == 1`:
`rx_hash_result = Title.rx_hash_result;`
 - Checksum: (For responder only)
 If `CQ.mini_cqe_res_format == 1`:
`checksum = mini_cqe_entry.checksum;`
 If `CQ.mini_cqe_res_format == 0`:
 HW checksum is not valid for the compressed cques (besides the first one that can be taken from title).
4. `Cqe_format`:
 The `cqe_format` field of the extracted CQE must be set to zero.
5. Consumer Counter Update:
 SW must increment the consumer counter for each extracted Mini CQE.

8.19.9.10CQE Owner Bit

HW updates the owner bit of the Title only after all the mini CQEs are written.

SW must update the owner bit in HW CQEs for all compressed CQEs except for the Title. HW CQE location is calculated using Consumer index- exactly the same way as location is calculated for the “ordinary” CQEs. Owner bit value should be taken from the Title.

Note: It is SW’s responsibility to make sure not to access the HW CQE location after Owner bit is updated. Also note that the Owner bit must be written before the DoorBell Record is updated.

8.19.9.11CQE Compressing and Interrupts

8.19.9.11.1EQE Generation

CQE compressing does not affect EQEs generation. That is to say when CQE compressing is enabled, EQEs are generated in the same way as when CQE compressing is disabled.

8.19.9.11.2CQE Compressing with CQ Moderation

Completion Event moderation mechanism treats each mini CQE as ordinary CQE. For example, if `CQ.cq_max_count == 4`, even if first 3 CQEs were compressed the fourth CQE arrival will cause EQE generation. In this case, compressing session is forcibly terminated, if needed, without waiting for compressing session timer expiration or occurrence of other condition. Please note that the fourth CQE will not be compressed.

When Completion Event Moderation timer expires before generating EQE, HW terminates CQE compressing session. Therefore, when CQE compressing is enabled together with Completion Event Moderation it does not “contribute” to the delay of CQE compressing expiration timeout.

8.19.9.12Limitations

- CQE compressing is not supported with Scatter to CQE
- CQE compression for 128 Byte cqe is supported only when `HCA_CAP.cqe_compression_128b==1`.
- CQE Signature is not supported for compressed CQEs besides the first compressed CQE in the session
- If `CQ.mini_cqe_res_format`, Checksum value report is not supported for compressed CQEs besides the first compressed CQE in the session.
- CQE compressing is not supported for LRO CQEs
- HW timestamp is not supported for compressed CQEs besides the first compressed CQE in the session.

8.19.10Completion Queue Context (CQC)

While opening a new CQ, software should create a CQ Context object and pass it to HW using `CREATE_CQ` command.

The format of a CQC structure is shown in Table 171.

Table 171 - Completion Queue Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
status				as notify	initiator_src_det	dbr_umem_valid		cqe_sz				cc	scqe break moderation en	oi	cq_period_mode		cqe compression en	mini_cqe_res_format		st													00h	
dbr_umem_id																		04h																
															page_offset														08h					
								log_cq_size							uar_page																0Ch			
								cq_period								cq_max_count												10h						
															c_eqn						14h													
								log_page_size																							18h			
																								1Ch										
								last_notified_index																20h										
								last_solicit_index																24h										
								consumer_counter																28h										
								producer_counter																2Ch										
local_partition_id															process_id												30h							
															thread_id												34h							
dbr_addr																		38h-3Ch																

Field descriptions are provided in Table 172.

Table 172 - Completion Queue Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	status	CQ status 0x0: OK 0x9: CQ_OVERFLOW 0xA: CQ_WRITE_FAIL 0xF: CQ_INVALID Valid for the QUERY_CQ. only when HCA_CAP.cq_invalidate==1, SW can set status to CQ_INVALID using MODIFY_CQ command.	
	27	as_notify	If set, AS Notify is enabled on this CQ. Supported only when HCA_CAP.as_notify==1. See TBD	
	26	initiator_src_dct	If set, initiator source dct is reported in CQE format instead of DCI. Supported only when HCA_CAP.initiator_src_dct_in_cqe==1. [Internal] For ConnectX-5 this field exposed only for EMC. All CQs must have the same configuration.	
	25	dbr_umem_valid	Indicates if dbr address is given directly or by by umem_id and umem_offset. When 0, dbr_addr allocates the physical address of the dbr. When 1, dbr_addr allocates the umem_offset of the dbr, and umem_offset is valid.	

Table 172 - Completion Queue Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:2 1	cqe_sz	CQE size 0x0: BYTES_64 0x1: BYTES_128_IF_NEEDED - For 128 bytes content of Cqe, 128 Bytes is generated. For 64 bytes content of Cqe, only the 64B of CQE content is generated. 0x2: BYTES_128_ALWAYS - For 128 bytes content of Cqe, size 128 Bytes is generated, same as mode 0x1. For 64 bytes content of Cqe, Zero padding of 64 Bytes will always be added before the 64B of CQE content. Supported only when HCA_CAP.cqe_128_always==1	
	20	cc	If set, all CQEs are written (collapsed) to first CQ entry	
	18	scqe_break_moderation_en	When set, solicited CQE (CQE.SE flag is enabled) breaks Completion Event Moderation. CQE causes immediate EQE generation. Supported only if HCA_CAP.scqe_break_moderation==1. For more info related to Completion moderation please refer to Section 8.20.19, "Completion Event Moderation," on page 508	
	17	oi	When set, overrun ignore is enabled. When set, updates of CQ consumer counter (poll for completion) or Request completion notifications (Arm CQ) DoorBells should not be rung on that CQ.	
	16:1 5	cq_period_mode	0: upon_event - cq_period timer restarts upon event generation. 1: upon_cqe - cq_period timer restarts upon completion generation. Supported only when HCA_CAP.cq_period_start_from_cqe==1.	
	14	cqe_compression_en	When set, CQE compressing feature is enabled for that CQ. For details See Section 8.19.9, "CQE Compressing," on page 459. For 128 BYTE CQE, can be set only if HCA_CAP.cqe_compression_128b==1.	
	13:1 2	mini_cqe_res_format	Mini Cqe responder format. valid only when cqe_compression==1. 0x0: resp_bytcnt_hash - Responder Mini CQE consists from: Byte Count and RX hash result. 0x1: resp_bytcnt_checksum - Responder Mini CQE consists from Byte Count and HW Checksum value. 0x3: resp_bytcnt_checksum_stride_index - Responder Mini CQE consists from Byte Count, HW Checksum and stride index. Supported only when mini_cqe_Resp_stride_index.	
	11:8	st	Event delivery state machine 0x6: SOLICITED_NOTIFICATION_REQUEST_ARMED 0x9: NOTIFICATION_REQUEST_ARMED 0xA: FIRED other: reserved Valid for the QUERY_CQ command only. Reserved for CREATE_CQ.	

Table 172 - Completion Queue Context Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	dbr_umem_id	UMEM ID of DB Record. Valid only when dbr_umem_valid==1. when valid, dbr_addr indicates umem_offset instead of physical address of the DB Record.	
08h	11:6	page_offset	Must be 0 The offset of CQ start in the first page allocated for the CQ buffer, in quantas of (page_size / 64)	
0Ch	28:24	log_cq_size	Log (base 2) of the CQ size (in entries). Maximum CQ size is 2 ²² CQEs (max log_cq_size is 22)	
	23:0	uar_page	UAR page this CQ can be accessed through (ringing CQ DoorBells)	
10h	27:16	cq_period	Event Generation moderation timer in 1 usec granularity, see See Section 8.20.19, "Completion Event Moderation," on page 508. 0 - CQ moderation disabled	
	15:0	cq_max_count	Event Generation Moderation counter, see See Section 8.20.19, "Completion Event Moderation," on page 508. 0 - CQ moderation disabled	
14h	7:0	c_eqn	EQ this CQ reports completion events to.	
18h	28:24	log_page_size	Log (base 2) of page size in units of 4KByte	
20h	23:0	last_notified_index	Last_notified_idx. Maintained by HW. Valid for QUERY_CQ command only. This field is for debug only purpose and is subject to change.	
24h	23:0	last_solicit_index	Solicit_producer_idx. Maintained by HW. Valid for QUERY_CQ command only. This field is for debug only purpose and is subject to change.	
28h	23:0	consumer_counter	Consumer counter. The counter is incremented for each CQE polled from the CQ. Must be 0x0 in CQ initialization. Indicates last consumer counter seen by HW (valid for the QUERY_CQ command only).	
2Ch	23:0	producer_counter	Producer Counter. The counter is incremented for each CQE that is written by the HW to the CQ. CQ overrun is reported if Producer_counter + 1 equals to Consumer_counter and a CQE needs to be added. Maintained by HW (valid for the QUERY_CQ command only)	
30h	31:20	local_partition_id	Valid only when as_notify==1	
	19:0	process_id	Valid only when as_notify==1	

Table 172 - Completion Queue Context Field Descriptions

Offset	Bits	Name	Description	Access
34h	15:0	thread_id	Valid only when as_notify==1	
38h-3Ch	64	dbr_addr	CQ DB Record physical address. Must be aligned to 8 bytes. Physical address of DB Record See 8.19.2 CQ DoorBell Record	

8.19.11CQ to EQ Remapping

The device supports a dynamic change of the EQ to which CQ reports Completion Events. This can be done via MODIFY_CQ command (See Section 29.9.4, “MODIFY_CQ – Modify CQ Parameters,” on page 1894). After successful completion of the command which remaps the EQ, the SW must call or schedule a procedure which polls the modified CQ.

8.19.12CQE Timestamping

Each Generated CQE includes a 64 bit timestamp. This field holds a sample of the device internal timer taken when the CQE was generated. The timestamp can be used by the application to monitor execution time of I/O operations.

CQEs for LRO operations coalescing TCP packets with the Timestamp Option do not carry the internal timer timestamp. This is indicated by the *lro_timestamp_is_valid* field in Cqe Format. See Table 145, “64B Completion Queue Entry Format Layout,” on page 435.

8.19.12.1Conversion to Real Time

The CQE timestamp represents an internal timer using device specific frequency. To translate the timestamp into real time, the ratio is given in *HCA_CAP.device_frequency_khz* field (See Table 1571, “HCA Capabilities Layout,” on page 1671). The timestamp should be divided by the frequency to produce a value in micro-seconds.

8.19.12.2Synchronization with Current Time

Presenting the actual time of CQE generation requires a reference point in time, synchronizing the wall-clock with the current device-clock. The current value of the device internal timer can be queried using a PCI read of the *Init_segment.internal_timer_h* and *Init_segment.internal_timer_l* fields (See Table 14, “Initialization Segment,” on page 234).

Note: Reading the internal timer using 2 PCI reads in a non-atomic manner may hit the wrap-around of the 32 LSBs of the timer, making the MSBs and LSBs incompatible. Reading the 32 MSBs twice (before and after reading the LSBs) can verify a wraparound did not happen.

8.20 Events and Interrupts

HCA has multiple sources that can generate events (completion events, asynchronous events/errors). Once an event is generated internally, it can be reported to the host software via the Event Queue mechanism. The EQ is a memory-resident circular buffer used by hardware to write event cause information for consumption by the host software. Once event reporting is enabled, event cause information is written by hardware to the EQ when the event occurs. If EQ is armed, HW will subsequently generate an interrupt on the device interface (send MSI-X message or assert the pin) as configured in the EQ.

Each one of the HCA EQs can be associated with a different event handler on the host. Each Event Group can be configured to report events to one of the EQs, implementing hardware demultiplexing of the events to different event handlers. In particular, completion reports can be reported to different EQs, based on the CQ that reports the event.

In virtual environment EQs can be exported to the guest VM, and kernel driver in Guest VM will control the EQ. HCA HW enforces protection and isolation between EQs.

SW can create new EQE using the GEN_EQE command.

8.20.1 Event Queues

The EQ is an object containing the following entities:

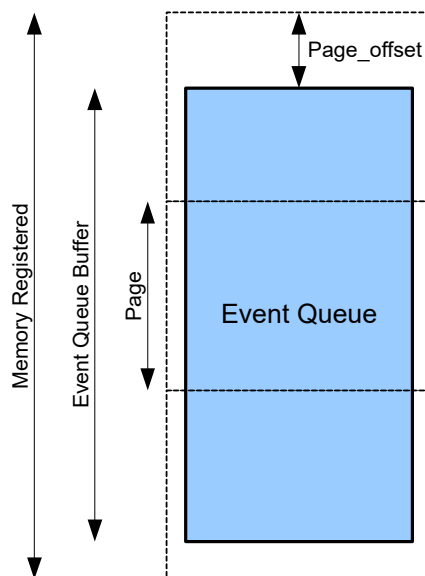
1. **Event Queue Context (EQC)** – The data structure describing the EQ properties that is passed to the device by SW while creating the EQ. The number of EQs supported by the hardware can be queried through the QUERY_HCA_CAP command (*log_max_eq*). The actual number of the EQs to be used by software (power of two) is configured at HCA initialization. EQC contains various parameters of the EQ. For more details see “Event Queue Context (EQC)” on page 505
2. **Event Queue** – A virtually-contiguous circular buffer accessible by system software and used by hardware to post event records - Event Queue Entries (EQEs). The maximum supported EQ size is retrieved by the QUERY_HCA_CAP command (*log_max_eq_sz*). EQE size is 64 bytes.
3. **DoorBell Register** associated with the EQ (see “UAR Page Format” on page 250).

8.20.2 Event Queue Buffer

An EQ is a virtually-contiguous memory buffer used by HCA to post event reports to and used by application software to poll events reports from. This buffer is allocated by the host software at EQ creation and its physical address is passed to the HCA. The memory associated with this buffer is passed to the HCA as a list of physical pages. EQ buffer must be aligned on EQE-size a boundary¹. EQ is shown in Figure 34.

1. If CPU cache size is larger than EQE-size, for best performance EQ buffer should be aligned on CPU cache line size and size of EQ buffer should be at least one CPU cache line.

Figure 34: Event Queue Buffer



Hardware posts event records (writing EQEs) to the EQ. Upon EQEs consumptions, SW updates the EQ Doorbell register specifying the Consumer Index and change state of the EQ (for example arm it to generate interrupt on next event posted).

EQ is a virtually-contiguous circular buffer accessible by the HCA hardware and contains EQEs. The buffer is managed by *Producer Counter* and *Consumer Counter* same as CQ is managed.

Upon polling the EQ successfully (that is, a new EQE is found), software updates the *Consumer Counter* which is used by hardware to monitor a potential EQ overflow.

If the EQ is full while an event (EQE) needs to be posted, it will be held off until SW polls EQEs and releases room for new EQEs to be posted. The number of completion events that can be held off is not limited by HW. However, if an EQ is full while other (non-completion events) need to be posted, they will be lost. It is SW responsibility to make sure that Event Queues used to report non-completion events are deep enough (or polled frequently enough) to avoid loss of events.

This EQ overrun handling must be fixed (if not fixed already)

8.20.2.1 EQE Ownership

EQE ownership is defined by **Owner** bit in the EQE. This bit follows the logic of the *Owner* bit in the CQE, for details refer to “CQE Ownership” on page 450.

8.20.2.2 EQE Format

Each EQE contains enough information to associate the EQE with its CQE. The EQE layout is shown in Table 173.

Table 173 - Event Queue Entry Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								event_type																event_sub_type								00h
																																04h-1Ch
event_data																																20h-38h
																signature												owner	3Ch			

EQE field descriptions are shown in Table 174.

Table 174 - Event Queue Entry Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:16	event_type	Event Type	
	7:0	event_sub_type	Event Sub Type. Defined for events which have sub types, zero elsewhere.	

Table 174 - Event Queue Entry Field Descriptions

Offset	Bits	Name	Description	Access
20h-38h	224	event_data	Delivers auxiliary data to handle the event. See Table 176, “Event_data Field - Completion Event Layout,” on page 479 See Table 178, “Event_data Field - DCT Events Layout,” on page 480 See Table 180, “Event_data Field - QP/SQ/RQ Events Layout,” on page 481 See Table 186, “Event_data Field - XRQ ERROR Events Layout,” on page 483 See Table 189, “Event_data Field - WQE Associated Page Fault Event Layout,” on page 484 See Table 191, “Event_data Field - RDMA Page Fault Event Layout,” on page 485 See Table 193, “Event_data Field - Completion Queue Error Event Layout,” on page 487 See Table 195, “Event_data Field - DROPPED_PACKET_LOGGED Event Layout,” on page 487 See Table 202, “Event_data Field - Port State Change Event Layout,” on page 491 See Table 205, “Event_data Field - GPIO Event Layout,” on page 492 See Table 207, “Event_data Field - Port Module Event Layout,” on page 492 See Table 209, “Event_data Field - DCBX Change Event Layout,” on page 493 See Section 211, “Event_data Field - PPS Event Layout,” on page 494 See Section 216, “Event_data Field - General Notification Delay Drop Timeout Event Layout,” on page 496 See Table 213, “Temperature Warning Event Layout,” on page 495 See Table 228, “Event_data Field - Doorbell / Blueflame Congestion Event Layout,” on page 500 See Table 231, “Event_data Field - Stall VL Event Field Descriptions,” on page 501 See Table 232, “Event_data Field - Command interface Completion Event Layout,” on page 501 See Table 234, “Pages Request Event Layout,” on page 502 See Table 236, “NIC Vport Change Event Layout,” on page 503 See Table 240, “Event_data Field - FPGA Error Event Layout,” on page 504 See Table 200, “Event_data Field - Driver Tracer Event Layout,” on page 489	

Table 174 - Event Queue Entry Field Descriptions

Offset	Bits	Name	Description	Access
3Ch	15:8	signature	Byte-wise XOR of EQE - signature protection (see See Section 17.2.4, “Completion and Event Queue Elements (CQEs and EQEs),” on page 879). EQE is valid if byte-wise XOR of entire EQE (including <i>signature</i> field) and the EQE index is 0xff.	
	0	owner	Owner of the entry	

While draining (consuming) the EQ, software should execute the following algorithm:

```
While (EQ[Consumer_indx].Owner == SW) {
    consume_entry();           // remove entry from the queue
    Consumer_counter++;
}
subscribe_for_event(EQ);      // subscribe for event for next time
```

8.20.2.3 EQ DoorBell Register

EQ DoorBell register is mapped to one of the UAR (DoorBell) page. Refer to “UAR Page Format” on page 250 for details.

8.20.3 Completion Events

HCA implements CQs which are described in detail in “Software Interface” on page 248. While creating a CQ, software configures the EQ number to which this CQ will report completion events.

Further description of completion events and requests for completion events notification is provided in “Request Completion Notification” on page 451.

8.20.4 Polling on EQEs

Some applications may use EQs without generating interrupts. This is useful when the interrupt rate is too high and polling makes sense to reduce the CPU load. In this case, the EQ should be created in fired state and never be armed. Further on, an EQ will be managed by polling the EQ and updating the EQ consumer index using DoorBells (see “UAR Page Format” on page 250).

8.20.5 Sharing MSI-X/Interrupt Amongst EQs

There are cases where more than one EQ report to the same MSI-X vector or to an interrupt pin. In such cases, when servicing an interrupt, the device driver should arm all the EQs that report to the same MSI-X vector/Interrupt pin.

8.20.6 Event Queue Mapping

Each event type can be mapped to an EQ. Mapping and unmapping events to/from EQs is performed through the command interface (see “Command Reference” on page 1647). Mapping completion events to EQs is performed separately for each CQ through the command interface (see “Command Reference” on page 1647).

The encoding for *event_type* fields is shown in Table 175, “Event Type and Coding”

Table 175 - Event Type and Coding

Family	Event Description	Event Type	EQE Format
Completion Events	Completion_Events	0x0	See Section 8.20.7, “Completion Events,” on page 479
Asynchronous Events	Path_Migrated_Succeeded	0x1	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Communication_Established	0x02	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Send_Queue_Drained	0x03	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Last_WQE_Reached	0x13	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	SRQ_Limit	0x14	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	DCT_All_Connections_Closed	0x1C	See Section 8.20.8.1, “DCT Events,” on page 480
	DCT_Access_Key_Violation	0x1D	See Section 8.20.8.1, “DCT Events,” on page 480
QP/RQ/SQ Affiliated Asynchronous Errors	CQ_Error	0x04	See Section 8.20.10.2, “Completion Queue Error Event,” on page 487
	Local_WQ_Catastrophic_Error	0x05	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Path_Migration_Failed	0x07	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Page_Fault_Event	0x0C	See Section 8.20.10.1, “Page Fault Event,” on page 483
	Invalid_Request_Local_WQ_Error	0x10	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Local_Access_Violation_WQ_Error	0x11	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	Local_SRQ_Catastrophic_Error	0x12	See Section 8.20.8.2, “QP/SQ/RQ/SRQ Events,” on page 480
	OBJECT_CHANGE_EVENT	0x27	See Section 8.20.9, “Object Change Event,” on page 481

Table 175 - Event Type and Coding

Family	Event Description	Event Type	EQE Format
Unaffiliated Asynchronous Events and Errors	Internal_Error	0x08	See Section 8.20.12.1, “Internal Errors,” on page 490.
	Port_State_Change	0x09	See Section 8.20.12.2, “Port State Change Event,” on page 491
	GPIO_Event TBD [Internal] Not supported.	0x15	See Section 8.20.12.3, “GPIO Event,” on page 492
	Port_Module_Event	0x16	See Section 8.20.12.4, “Port Module Event,” on page 492
	Temp_Warning_Event	0x17	See Section 8.20.12.7, “Temp Warning Event,” on page 495
	Remote_Configuration_Protocol_Event	0x19	
	DCBX_Change_Event	0x1E	See Section 8.20.12.5, “DCBX Change Event,” on page 493
	PPS_Event	0x25	See Section 8.20.12.6, “PPS Event - Pulse Per Second Event,” on page 494
	General_Notification_Event	0x22	See Section 8.20.12.8, “General Notification Event,” on page 495
	Monitor_Counter_Change_Event	0x24	See Section 8.20.12.9, “Monitor Counter Change Event,” on page 499
Performance related events	Doorbell_Blueflame_Congestion_Event	0x1A	See Section 8.20.13.1, “Doorbell / Blueflame Congestion Event,” on page 499
	Stall_VL_event	0x1B	See Section 8.20.13.2, “Stall VL Event,” on page 500
Debug events	DROPPED_PACKET_LOGGED_Event	0x1F	See Section 8.20.11.1, “DROPPED_PACKET_LOGGED Event,” on page 487
	Resource_Dump_Event	0x28	See Section 8.20.11.3, “Resource Dump Event,” on page 489
	Device_Tracer_Event	0x26	See Section 8.20.11.2, “Driver Tracer Event,” on page 488

Table 175 - Event Type and Coding

Family	Event Description	Event Type	EQE Format
HCA Inter-face	Command_Interface_Com- pletion	0x0A	See Section 8.20.14.1, “Command Interface Com- pletion Event,” on page 501
	Page_Request	0x0B	See Section 8.20.14.2, “Pages Request Event,” on page 501
	Nic_Vport_Change	0x0D	See Section 8.20.14.3, “NIC Vport Change Event,” on page 502
	ESW_FUNC- TIONS_CHANGED	0x0E	
FPGA	FPGA_Error	0x20	See Section 8.20.15.1, “FPGA Error Event,” on page 504
	FPGA_QP_Error	0x21	See Section 8.20.15.2, “FPGA QP Error Event,” on page 504
XRQ	XRQ_ERROR	0x18	See Section 8.20.10, “XRQ Error Event,” on page 482

The *Event_Data* field written to the EQ is defined in the following sections.

8.20.7 Completion Events

EQE event data is used for reporting completion events. Its layout is shown in Table 176.

Table 176 - Event_data Field - Completion Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-14h
																cq_number																18h

Table 177 - Event_data Field - Completion Event Field Descriptions

Offset	Bits	Name	Description	Access
18h	23:0	cq_number		

8.20.8 Asynchronous Events and Errors

8.20.8.1 DCT Events

The following is an event format generated for DCT_All_Connections_Closed and DCT_Access_Key_Violation.

Table 178 - Event_data Field - DCT Events Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																00-14h	
									dctn																								18h

Table 179 - Event_data Field - DCT Events Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:24	reserved		
	23:0	dctn	DC Number	

8.20.8.2 QP/SQ/RQ/SRQ Events

The EQE event data format shown in Table 180 on page 481 is used for the following QP/SQ/RQ/SRQ related events:

- Path Migration Succeeded
- Path Migration Failed
- Communication Established
- Send Queue Drained
- Local WQ Catastrophic Error
- Invalid Request Local WQ Error
- Local Access Violation WQ Error
- SRQ Limit Reached
- Last WQE Reached Event
- Local SRQ Catastrophic Error
- Spoof protection failure

•

Table 180 - Event_data Field - QP/SQ/RQ Events Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00-10h
type																																14h
								qpn_or_rqn_or_sqn																								18h

Table 181 - Event_data Field - QP/SQ/RQ Events Field Descriptions

Offset	Bits	Name	Description	Access
14h	31:2 4	type	Queue type 0: QP 1: RQ 2: SQ	
18h	23:0	qp_n_or_rqn_or_sqn	Indicates QPN, SQN or RQN depending on type field.	

For the “Local Access Violation WQ Error”, “Local WQ Catastrophic Error”, “Invalid Request Local WQ Error” and “Local SRQ Catastrophic Error” event types - the event subtype denotes the Mellanox specific error syndrome. For further information contact Mellanox technical support.

8.20.9 Object Change Event

This event is used to report change in an object.

Upon receiving this event, SW should query the changed object to get the new one.

Note that after receiving an event, object becomes in Fired state, and only after performing QUERY command, it returns to Armed state. Without performing QUERY, object remains in Fired state and no new event will be generated.

Table 182 - Event_data Field - Object Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
Affiliated Event Header (See Table 184, “Affiliated Event Header Layout,” on page 482)																																00h-04h
																																08-18h

Table 183 - Event_data Field - Object Change Event Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	Affiliated Event Header (See Table 184, “Affiliated Event Header Layout,” on page 482)	See Table 184, “Affiliated Event Header Layout,” on page 482	

Table 184 - Affiliated Event Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																obj_type																00h	
obj_id																																	04h

Table 185 - Affiliated Event Header Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	obj_type	Object Type. Enumeration of object types is described in HCA_CAP.general_obj_types field.	
04h	31:0	obj_id	Object Id	

8.20.10XRQ Error Event

Note: This section is preliminary and subject to change.

XRQ may induce an event upon error. Some of the event may carry the data associated within the events while other may require the software to query the parameters using the `xrq_error_handle` communicated in the event. It is highly recommended to query the parameters of the Queried Error as soon as possible to prevent the HCA from the need to store them.

Table 186 - Event_data Field - XRQ ERROR Events Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00-10h
type								xrqn																								14h
								qpn_or_backend_controller_id_or_xrq_error_handle																								18h

Table 187 - Event_data Field - XRQ ERROR Events Field Descriptions

Offset	Bits	Name	Description	Access
14h	31:24	type	Error type 0x0: QP_ERROR - Error induced by transport communication on a QP associated with this XRQ. QP number is reported in qpn field. 0x1: BACKEND_CONTROLLER_ERROR - Error induced by the backend controller associated with this XRQ. backend_controller identifier is reported in backend_controller_id field. 0x2: DC_ERROR - Error induced by DC communication associated with this XRQ.	
	23:0	xrqn		
18h	23:0	qpn_or_backend_controller_id_or_xrq_error_handle	For type==QP_ERROR - indicates the QP number on which the error occur. For type==BACKEND_CONTROLLER_ERROR - indicates the backend_controller_id number on which the error occur. For type==DC_ERROR - indicates the XRQ error handle. SW should use this handle in QUERY_XRQ_ERROR_PARAMS commands to get the error parameters.	

8.20.10.1Page Fault Event

Page fault event indicates that the QP encountered a page fault. The QP will have at most four concurrent page fault events: Requester-Read, Requester-Write, Responder-Read, and Responder-Write. When the relevant page fault is handled and corrected by software, it should indicate this to the HCA using PAGE_FAULT_RESUME command (See Section 29.6.4, “PAGE_FAULT_RESUME,” on page 1869)

The event subtype denotes whether the page fault occurred due to WQE or RDMA operation as described in Table 188.

Table 188 - Page Fault Event Subtype

Subtype Encoding	Event Subtype	Description
0x0	WQE page fault	The EQE describes a WQE page-fault and is formatted according to Table 189, “Event_data Field - WQE Associated Page Fault Event Layout,” on page 484.
0x1	RDMA page fault	The EQE describes an RDMA page-fault event and is formatted according to Table 191, “Event_data Field - RDMA Page Fault Event Layout,” on page 485.

Table 189 - Event_data Field - WQE Associated Page Fault Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bytes_committed																															00h	
																wqe_index																04h
																len																08h
									token																							0Ch
																															10h-14h	
						page_- fault_type		wq_number																								18h

Table 190 - Event_data Field - WQE Associated Page Fault Event Field Descriptions

Offset	Bits	Name	Description
00h	31:0	bytes_committed	The number of bytes already sent/received in this WQE before the fault.
04h	15:0	wqe_index	The WQE index (from beginning of the relevant queue, in WQEBB units) where the fault occurred

Table 190 - Event_data Field - WQE Associated Page Fault Event Field Descriptions

Offset	Bits	Name	Description
08h	15:0	len	Length of memory access operation that experienced a page fault. For page faults on send operations in the responder side we need to know how much of the WQE's memory we must fetch successfully. We may try to fetch more of the WQE's memory, beyond the needs of the current packet, but a failure to do that will not cause a QP error.
0Ch	23:0	token	A unique number identifying the origin of the page fault. Must be provided unchanged to the PAGE_FAULT_RESUME command.
18h	26:2 4	page_fault_type	Page fault type. 0x0: RMP related page fault 0x1: Requester QP WQE pagefault on send or write opcodes 0x2: Responder QP receive WQE pagefault, no SRQ involved 0x3: Requester QP WQE pagefault on read or atomic opcodes 0x4: RDMA Read responder pagefault (not valid). 0x5: Tag matching related pagefault 0x6: RDMA write responder pagefault (not valid). 0x7: SQ related page fault
	23:0	wq_number	The work queue identifier for the work queue on which the page fault happened. For page_fault_type == 0, indicates the rmp number. For page_fault_type == 1,2,3 indicates the QP or DCI number. For page_fault_type == 5 indicate the XRQ number. For page_fault_type == 7 indicates the the SQ number. Must be provided in the PAGE_FAULT_RESUME command.

Page fault events of subtypes of RDMA-Read Responder and RDMA-Write Responder create an event with the data format described in Table 191, "Event_data Field - RDMA Page Fault Event Layout".

Table 191 - Event_data Field - RDMA Page Fault Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bytes_committed																															00h	
r_key																															04h	
																packet_len																08h
rdma_op_len																															0Ch	
rdma_va[63:32]																															10h	
rdma_va[31:0]																															14h	

Table 191 - Event_data Field - RDMA Page Fault Event Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
					page - fault_type			token																								18h

Table 192 - Event_data Field - RDMA Page Fault Event Field Descriptions

Offset	Bits	Name	Description
00h	31:0	bytes_com- mitted	The number of bytes already sent/received in this WQE before the fault.
04h	31:0	r_key	The R-key used for the RDMA transaction which caused the fault.
08h	15:0	packet_len	The length of the packet received/constructed for transmission when the fault occurred. Measured in bytes. This can be used on RDMA write operations, where we do not always know the rdma_op_len. The software may attempt to prefetch memory based on the event attributes, but must make sure it can map the pages needed for packet_len bytes.
0Ch	31:0	rdma_op_len	The length of the RDMA operation, in bytes. If the HW cannot reliably report the length, it will put 0 in this field, and the driver should perform prefetching based on access pattern
10h	31:0	rdma_va[63:32]	The VA used for the beginning of the RDMA operation, upper 32 bits
14h	31:0	rdma_va[31:0]	The VA used for the beginning of the RDMA operation, lower 32 bits
18h	26:24	page_fault_- type	Page fault type. 0x0: RMP related page fault (not valid). 0x1: Requester QP WQE pagefault on send or write opcodes (not valid). 0x2: Responder QP receive WQE pagefault, no SRQ involved (not valid). 0x3: Requester QP WQE pagefault on read or atomic opcodes (not valid). 0x4: RDMA Read responder pagefault. 0x5: Tag matching related pagefault (not valid). 0x6: RDMA write responder pagefault 0x7: SQ related page fault (not valid).
	23:0	token	A unique number identifying the origin of the page fault. Must be provided unchanged to the PAGE_FAULT_RESUME command

8.20.10.2 Completion Queue Error Event

The following EQE event data format is used for reporting CQ related errors.

Table 193 - Event_data Field - Completion Queue Error Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
								cqn																													00h
																																04h					
																								syndrome								08h					
																																0Ch-18h					

Table 194 - Event_data Field - Completion Queue Error Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:0	cqn	CQ number event is reported for	
08h	7:0	syndrome	0x1: CQ_overnrun 0x2: CQ_access_violation_error	

8.20.11 Debug Events

8.20.11.1 DROPPED_PACKET_LOGGED Event

This event indicates that a dropped packet head was logged in the device. The packet head can be retrieved by the GET_DROPPED_PACKET_LOG command. See “Dropped Packet Logging” on page 1049 for details.

Table 195 - Event_data Field - DROPPED_PACKET_LOGGED Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-18h

Table 196 - Event_data Field - DROPPED_PACKET_LOGGED Event Field Descriptions

Offset	Bits	Name	Description	Access
00h-18h	256	reserved		

8.20.11.2 Driver Tracer Event

Table 197 - Event_data Field - Driver Tracer Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
reserved																																00h-18h

Table 198 - Event_data Field - Driver Tracer Event Field Descriptions

Offset	Bits	Name	Description	Access
00h-18h	256	reserved		

This event is used by the Device Tracer for two types of notifications:.

Table 199 - Driver Tracer Event Subtype

Subtype Encoding	Event Subtype	Description
0x0	LOST_OWNERSHIP	Changes in Device Tracer ownership
0x1	BUFFER_CHANGE	Traces may have been logged in the Tracer buffer since last BUFFER_CHANGE event

8.20.11.3 Resource Dump Event

Table 200 - Event_data Field - Driver Tracer Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
vhca_id_valid																segment_type														00h		
																														04h		
																vhca_id														08h		
																														0Ch		
		num_of_obj1														num_of_obj2														10h		
																														14h-18h		

Table 201 - Event_data Field - Driver Tracer Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	29	vhca_id_valid	If set, then vhca_id field is valid. Otherwise dump resources on my vhca_id.	
	15:0	segment_type	See Section 27.8, “Resource Dump,” on page 1062.	
04h	15:0	vhca_id	vhca_id where the resource is allocated.	
08h	31:0	index1	First object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	
0Ch	31:0	index2	Second object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	

Table 201 - Event_data Field - Driver Tracer Event Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
10h	31:16	num_of_obj1	The amount of objects to dump starting for index 1. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj1_supports_all is set, the special value of 0xffff represents "all". When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents "active". The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj1 is "0".	
	15:0	num_of_obj2	The amount of objects to dump starting for index 2. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj2_supports_all is set, the special value of 0xffff represents "all". When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents "active". The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj2 is "0".	

8.20.12 Unaffiliated Events and Errors

8.20.12.1 Internal Errors

Internal Errors are reported to a buffer in the HCA address space. An EQE is not generated in the case of an internal error. The device driver can either periodically poll on the buffer or register to receive an interrupt in case of an internal error.

Internal error buffer is located on the initialization segment of the HCA BAR. Interrupt can be generated regardless whether the EQ is armed. Under certain conditions, an interrupt will not be generated; it is therefore recommended to periodically poll on the Internal Error buffer.

When an error is reported, the Internal Error buffer will cause a non zero value to appear in the first DWORD of the buffer. When software detects an error indication, that is, a valid buffer, it should log the entire buffer for debug purposes.

Internal errors can be either reported by the device or detected by the device driver. Errors detected by the device can be for example: unrecognized EQE, bus error when reading from device configuration space, etc. When an error is detected, hardware reset should be performed.

8.20.12.2 Port State Change Event

The following EQE event data format is used for reporting port state change.

Table 202 - Event_data Field - Port State Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
																																00h-04h				
port_num																																				08h
																																0Ch-18h				

Table 203 - Event_data Field - Port State Change Event Field Descriptions

Offset	Bits	Name	Description	Access
08h	31:28	port_num	Port number	

The event subtype denotes whether the port new state is up or down, as shown in Table 204.

Table 204 - Port State Change Event Subtype

Subtype Encoding	Event Subtype	Description
0x1 0x4 0x5	port state changed	port changed its state.
0x1	port Link Down	Logical link of the port is down.
0x4	port Link Active	Logical link of the port is active.
0x5	port Link Initialized	Logical link of the port is at Initialized state
0x6	Lid changed	port LID changed
0x7	PKEY changed	port PKey table changed
0x8	GUID changed	port GUID changed
0x9	client reregister	Client Reregister

8.20.12.3 GPIO Event

The HCA can be configured to generate an event upon a (rising/falling) change of one or more GPIO input pins. Changes on multiple GPIO pins may be reported using a single EQE.

Note that it is guaranteed that a change on a GPIO input will cause an event generation. However, if the rising/falling rate is too high, multiple changes may be collapsed and reported by a single EQE.

Table 205 provides the EQE event data format used for reporting GPIO events. Field descriptions follow in Table 205.

Table 205 - Event_data Field - GPIO Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-08h
gpio_event_hi																																0Ch
gpio_event_lo																																10h
																																14-18h

Table 206 - Event_data Field - GPIO Event Field Descriptions

Offset	Bits	Name	Description	Access
00h-08h	96	reserved		
0Ch	31:0	gpio_event_hi	If any bit is set to '1', then a rising/falling event has occurred on the corresponding GPIO pin.	
10h	31:0	gpio_event_lo	If any bit is set to '1', then a rising/falling event has occurred on the corresponding GPIO pin.	

8.20.12.4 Port Module Event

The Port Module Event reports the module plug and unplug state change of a module. A change in the module's state generates an event when a change happens.

This event is supported only when HCA_CAP.port_module_event==1.

Table 207 - Event_data Field - Port Module Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
									module																	module_status		00h				
																					error_type										04h	

Table 207 - Event_data Field - Port Module Event Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-18h

Table 208 - Event_data Field - Port Module Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:16	module	Module Number. SW can use PMLP register to know port number to module number mapping.	
	3:0	module_status	Module Status. 0x1: Plugged 0x2: Unplugged 0x3: Module plugged with Error- Details in errors reported in error_type field.	
04h	11:8	error_type	Module error type. Valid when module_status==0x3 0x0: Power Budget Exceeded 0x1: Long Range for non MLNX cable/module 0x2: Bus stuck(I2C or data shorted) 0x3: No EEPROM/retry timeout 0x4: Enforce part number list 0x5: Unknown identifier 0x6: High Temperature 0x7: Cable is shorted <u>0x8: PMD_type_is_not_enabled - (see PMTPS).</u> <u>0x9: Laster TEC failure</u> <u>0xa: High current</u> <u>0xb: High voltage</u> <u>0xc: pcie_system_power_slot Exceeded</u> <u>0xd: High power</u> <u>0xe: Module_state_machine_fault</u>	

8.20.12.5DCBX Change Event

This event is generated to report and change in DCBX state or configuration.

Table 209 - Event_data Field - DCBX Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				error							event_sub_- type			00h		
																																04h-18h

Table 210 - Event_data Field - DCBX Change Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	11:8	error	Indicates the error cause the event. valid only when event_sub_type==ERROR_STATE.	
	3:0	event_sub_type	Event subtype. 0x1: ERROR_STATE – DCBX has reached an error state. The error is reported in the error field. The error enumeration is according to its definition in error field in the DCBX_PARAM register. 0x2: REMOTE_CONFIG_CHANGE - Change in remote party config. New parameters reported by PFCC, QTCT, QETCR and DCBX_PARAM registers for remote config change. 0x3: LOCAL_OPER_CHANGE - Change in local party operational state. New parameters reported by PFCC, QTCT, QETCR and DCBX_PARAM registers for local operational change 0x4: REMOTE_CONFIG_APPLICATION_PRIORITY_CHANGE - Change in DCBX_APPLICATION register. No event data is attached.	

8.20.12.6 PPS Event - Pulse Per Second Event

This event reports PPS in and out changes. This event supported only when HCA_CAP.pps ==1.

For more information regarding using this event please refer to Section 13.3.1.1, “Auxiliary Pulse-Per-Second (PPS) Signals,” on page 833.

Table 211 - Event_data Field - PPS Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								pin				00h				
																																04h
time_stamp_h																												08h				
time_stamp_l																												0Ch				
																																10h-18h

Table 212 - Event_data Field - PPS Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	7:0	pin	Indicating the pin where PPS event was generated	
08h	31:0	time_stamp_h	MSB of the device time when it was triggered from the external GPIO pin. In_time for ptp_in, and out_time for ptp_out. Time is in units of device clock. Device clock frequency can be fetched from HCA_CAP.device_frequency	
0Ch	31:0	time_stamp_l	MSB of the device time when it was triggered from the external GPIO pin. In_time for ptp_in, and out_time for ptp_out. Time is in units of device clock. Device clock frequency can be fetched from HCA_CAP.device_frequency	

8.20.12.7Temp Warning Event

This event supported only when HCA_CAP.temp_warn_event==1.

This event used to report for over temperature warning. SW should use MTCAP and MTMP registers to configure the warning event parameters.

Note that this event will not be reported for temperature warning issues happened before driver registering to this event.

Table 213 - Temperature Warning Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sensor_warning																															00h-0Ch	

Table 214 - Temperature Warning Event Field Description

Offset	Bits	Name	Description	Access
00h-0Ch	128	sensor_warning	Bit vector indicating which of the sensor reading is above threshold. Address 00h bit31 is sensor_warning[127] Address 0Ch bit0 is sensor_warning[0]	RO

8.20.12.8General Notification Event

General notifications event is issued by the device for a variety of non-fatal occurrences. Driver handling for this event is optional, and the device may resume functioning in its current state. The driver is recommended to handle correctly the event according to the specified *event_sub_*

type. In case of an unknown *event_sub_type* driver must ignore the event (except for notifying of the event and its content for debug purposes).

The event is supported when HCA_CAP.general_notification_event==1.

Table 215 - General Notification Event Subtype

Subtype Encoding	Event Subtype	Description
0x1	Delay Drop Timeout	Timeout expired for a packet waiting for receive WQE posting. See Section 8.14.3, “Drop-Less RQ,” on page 398
0x2	Receive Hang	
0x3	Transmit Engine Hang	See Section 27.6, “Device Health Events,” on page 1061
0x4	Power Failure Event	A notification of a power failure.
0x5	PCI_POWER_CHANGE_EVENT	This event is generated when the status of pci power is changed. When SW gets this event, it should query MPEIN Register to get the updated status and power.
0x6	Pcie_Hanged_Event	See Section 27.7, “PCIe Hanged Event,” on page 1061
0x7	LIVE_FW_PATCH	This event is generated upon live FW patch
0x8	PCI_SYNC_FOR_FW_UPDATE	Supported only when HCA_CAP.pci_sync_for_fw_update_event==1. See Table 224, “Event_data Field - General Notification PCI SYNC FOR FW UPDATE Event Layout,” on page 498
0x9	Device Buffer Update	Indicates that the buffer configuration/ownership was changed. SW should query PFCC pfcc.buf_ownership to get the updated info.

Table 216 - Event_data Field - General Notification Delay Drop Timeout Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
								rq_user_index_delay_drop																													00h
																																04h-18h					

Table 217 - Event_data Field - General Notification Delay Drop Timeout Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:0	rq_user_index_delay_drop	The User Index of the RQ that cause the timeout	

Table 218 - Event_data Field - General Notification Transmit Engine Hang Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																vhca_id												00h				
transmit_engine_id								qpn																					04h			
																																08h-18h

Table 219 - Event_data Field - General Notification Transmit Engine Hang Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	vhca_id	vhca_id	
04h	31:24	transmit_engine_id		
	23:0	qpn		

Table 220 - Event_data Field - General Notification PCI Power Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-18h

Table 221 - Event_data Field - General Notification PCI Power Change Event Field Descriptions

Offset	Bits	Name	Description	Access

Table 222 - Event_data Field - General Notification PCIe Hanged Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																host_hanged_bitmask												00h				
																																04h-18h

Table 223 - Event_data Field - General Notification PCIe Hanged Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	host_hanged_bitmask	BitMask indicates which hosts that it's pcie is hanged.	

Table 224 - Event_data Field - General Notification PCI SYNC FOR FW UPDATE Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												pci_sync_for_fw_update_rst_state		00h		
																														04h-18h		

Table 225 - Event_data Field - General Notification PCI SYNC FOR FW UPDATE Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	3:0	pci_sync_for_fw_update_rst_state	PCI synced reset State. 0x0: RESET_REQUEST - Indicates that a synced reset is requested 0x1: RESET_NOW - Indicates to drivers to toggle the PCIe link. 0x2: RESET_ABORT - Sync reset flow did not complete successfully. Reset did not happen	RO

8.20.12.9 Monitor Counter Change Event

This event is used to notify on a change of monitored counters using SET_MONITOR_COUNTER (see Section 29.35.4, “SET_MONITOR_COUNTER - Set Monitor Counter,” on page 2195).

Table 226 - Monitor Counter Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch

Table 227 - Monitor Counter Change Event Field Description

Offset	Bits	Name	Description	Access
0h-0Ch	128			

8.20.13 Performance Related Events

8.20.13.1 Doorbell / Blueflame Congestion Event

If the rate of DoorBell or BlueFlame ringing is higher than the rate which the device can handle, the on-chip DoorBell and BlueFlame resources become congested. This may have a performance

and QoS impact. In order for the SW to mitigate the congestion, the device can issue an event that indicates that DoorBells and Blue Flames resources are congested.

Table 228 - Event_data Field - Doorbell / Blueflame Congestion Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
event_subtype																congestion_level																00h
																																04h
																																08h
																																0Ch
																																10h
																																14h

Table 229 - Event_data Field - Doorbell / Blueflame Congestion Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	event_subtype	0 - Doorbell congestion 1 - BlueFlame congestion	
	15:8	congestion_level	0 - N/A 1 - Mild congestion ... 7 - Severe congestion	

8.20.13.2Stall VL Event

When a VL is unable to transmit packets for some time because of lack of credits, it will move to “Stall mode”. The Stall VL Event informs SW of this state.

Table 230 - Event_data Field - Stall VL Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																							port_num					vl			00h	
																																04h
																																08h
																																0Ch
																																10h
																																14h

Table 231 - Event_data Field - Stall VL Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	7	port_num	port number	
	3:0	vl	VL number	

8.20.14HCA Interface Events

8.20.14.1Command Interface Completion Event

Table 232 provides the EQE event data format used for reporting command completion. HCR (HCA Command Register) fields which are included in the EQE are explained in detail in “Command Interface” on page 613.

Table 232 - Event_data Field - Command interface Completion Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
command_completion_vector																																00h
																																04-18h

Table 233 - Event_data Field - Command Interface Completion Event Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	command_completion_vector	Bits in this vector are set for commands completed and not yet reported by event	

8.20.14.2Pages Request Event

This event is used to inform the driver that the device needs to add / release pages for one of its functions. In response to such an event, the driver will post a `MANAGE_PAGES` command. `MANAGE_PAGES` is executed once, per event, (per `function_id`) in response to Page Request Event. This command also acts as arming the Page Request Event. This implies that once Page Request Event is generated for a `function_id`, it will not be generated again before the driver exe-

cludes the `MANAGE_PAGES` command. It is recommended for the PF driver to have place in the EQ for such an event per VF associated with it.

Table 234 - Pages Request Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
embedded cpu function		release all pages														function_id												00h				
																												04h				
num_pages																																08h-18h

Table 235 - Pages Request Event Field Descriptions

Offset	Bits	Name	Description
00h	31	embedded_cpu_function	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU HCA - CAP.embedded_cpu==1
	30	release_all_pages	If set, SW should release all pages for the reported function_id without calling manage pages command on that function. Supported only when HCA_CAP.release_all_pages==1.
	15:0	function_id	Which function requested to claim or reclaim pages 0 - my hca. others - function_id of specific vhca. used by vhca_group_manager(HCA_CAP.vhca_group_manager==1).
04h	31:0	num_pages	Number of missing / unneeded pages (signed number, msb indicate sign). This is just a recommendation for the driver. the actual number of pages that the driver deliver to the device is as set in the <code>MANAGE_PAGES</code> command. NA when <code>release_all_pages==1</code> .

8.20.14.3 NIC Vport Change Event

This event is used to inform the e-switch manager that the one of the `nic_vport_contexts` had been changed.

For more details on using this event, please refer to Section 14.1.6, “NIC Vport Change Notification,” on page 844

Table 236 - NIC Vport Change Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
																vport_num												00h						
																																		04h-18h

Table 237 - NIC Vport Change Event Field Descriptions

Offset	Bits	Name	Description
00h	15:0	vport_num	Which function requested to claim or reclaim pages 0 - my hca. others - function_id of specific vhca. used by vhca_group_manager(HCA_CAP.vhca_group_manager==1).

8.20.14.4ESW_FUNCTIONS_CHANGED Event

Table 238 - ESW_FUNCTIONS_CHANGED Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-18h

Table 239 - ESW_FUNCTIONS_CHANGED Event Field Descriptions

Offset	Bits	Name	Description

8.20.15 FPGA Events

8.20.15.1 FPGA Error Event

This event is used to report on FPGA internal errors to software.

Table 240 - Event_data Field - FPGA Error Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00-04h
																								syndrome								08h
																																0Ch-18h

Table 241 - Event_data Field - FPGA Error Event Field Descriptions

Offset	Bits	Name	Description	Access
08h	7:0	syndrome	0x1: corrupted_ddr 0x2: flash_timeout 0x3: internal_link_error 0x4: watchdog_failure 0x5: I2C_failure 0x6: Image_changed - Doesn't necessarily indicate error event. It can also indicate the end of a successful flow. SW needs to read the status in FPGA_CTRL. 0x7: high_temperature	

8.20.15.2 FPGA QP Error Event

This event is used to report on FPGA QP errors to software.

Table 242 - Event_data Field - FPGA QP Error Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00-04h
																								syndrome								08h
																																0Ch-14h
																fpga_qpn																18h

Table 243 - Event_data Field - FPGA QP Error Event Field Descriptions

Offset	Bits	Name	Description	Access
08h	7:0	syndrome	0x1: retry_counter_expired 0x2: RNR_expired	
18h	23:0	fpga_qpn	FPGA QP Number	

8.20.16Event Queue Context (EQC)

The number of EQs supported by the HCA can be retrieved by QUERY_HCA_CAP command.

Note: EQ enumeration is per function in a multi-function device. Each one of the functions owns EQs numbered from 0x0. When an EQ is accessed, it is identified by both EQ number.

Table 244 - Event Queue Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status													ec	oi						st												00h
																																04h
																				page_offset												08h
				log_eq_size				uar_page																								0Ch
																																10h
																				intr												14h
				log_page_size																												18h
																																1Ch
																																20h-24h
								consumer_counter																								28h
								producer_counter																								2Ch
																																30h-3Ch

EQC fields are shown in Table 245.

Table 245 - Event Queue Context Field Descriptions

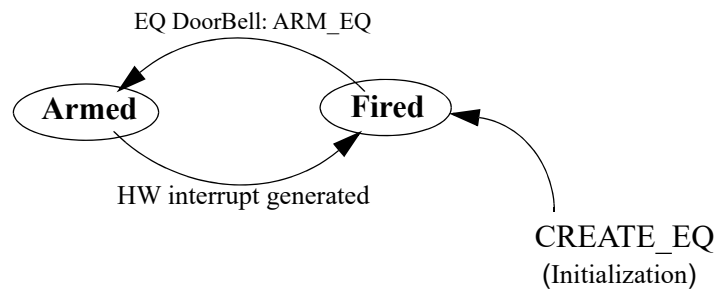
Offset	Bits	Name	Description	Access
00h	31:28	status	EQ status 0x0: OK 0xA: EQ_WRITE_FAILURE Valid for the QUERY_EQ command only	
	18	ec	Is set, all EQEs are written (collapsed) to first EQ entry	
	17	oi	When set, overrun ignore is enabled.	
	11:8	st	Event delivery state machine 0x9: ARMED 0xA: FIRED Reserved for CREATE_EQ.	
08h	11:6	page_off set	Offset bits[11:5] of first EQE in the EQ relative to the first page in memory region mapping this EQ, in quantas of (page_size / 64) Must be 0	
0Ch	28:24	log_eq_ size	Log (base 2) of the EQ size (in entries).	
	23:0	uar_page	UAR page this EQ can be accessed through (ringing EQ DoorBells)	
14h	711:0	intr	MSI-X table entry index to be used to signal interrupts on this EQ. Reserved if MSI-X is not enabled in the PCI configuration header.	
18h	28:24	log_page_ size	Log (base 2) of page size in units of 4KByte	
28h	23:0	consum- er_count er	Consumer counter. The counter is incremented for each EQE polled from the EQ. Must be 0x0 in EQ initialization. Indicates last consumer counter seen by HW (valid for the QUERY_EQ command only).	
2Ch	23:0	produc- er_count er	Producer Counter. The counter is incremented for each EQE that is written by the HW to the EQ. EQ overrun is reported if Producer_counter + 1 equals to Consumer_counter and a EQE needs to be added. Maintained by HW (valid for the QUERY_EQ command only) Should be 0x0 in EQ initialization.	

8.20.17 Hardware Interrupts

Reporting an event to the EQ can be accompanied by the generation of a hardware interrupt (generating an interrupt message on the host interface bus). Each EQ can be independently configured to assert one of the interrupt pins or generate one of the Interrupt Messages (MSI-X) on the host interface bus.

A hardware interrupt is generated if the EQ has been armed by Interrupt Handling software. While EQ is in **Armed** state, when an event is generated, hardware disarms the EQ and no additional interrupts are generated until the EQ is re-armed. The EQ state machine and state transitions are shown in Figure 35, “EQ State Machine”.

Figure 35: EQ State Machine



Arming an EQ to trigger an interrupt is done by writing an EQ ARM DoorBell. Refer to “UAR Page Format” on page 250.

MSI-X is generated only if the MSI-X enable bit is set in the appropriate PCI configuration register of the device. According to the PCI specification, when MSI-X is enabled, a device is prohibited from using the INTA# pin. Therefore, EQs must be configured in accordance with the MSI-X enable bit in PCI configuration space to avoid a conflict with the PCI specification. When a device is reconfigured to a different interrupt scheme (from MSI-X to INTA# or vice versa), the EQs should be torn down and re-established with the new configuration.

[In addition, HCA provides mechanism to moderate interrupt requests' generation by the event queues as detailed in “Interrupt Moderation” on page 507.](#)

8.20.18 Interrupt Moderation

The HCA supports hierarchical event delivery scheme. The completion of Work Request on WQ is reported to CQ. Multiple WQs can report completions to the same CQ. CQ that is armed to generate event upon Work Request completion posts event to one of the EQs. Multiple CQs can report an event to same EQ. An EQ that is armed to generate interrupt upon Event report, generates interrupt by the mean configured (for example specific MSI-X message, certain pin assertion etc.). Multiple EQs can generate interrupts by the same mean (for example by generating same MSI-X message or asserting same pin).

Under high load of network traffic, it is highly desirable to moderate event reporting, hereby enabling the SW to handle multiple events upon each entry to the event handler.

HCA deploys a two-stage moderation scheme:

1. **Completion Event Moderation** - CQs can be configured to hold off event generation (posting event report to the EQ) until certain threshold of completions are reported to that CQ (or a timeout is reached) while it is armed. Each CQ can be independently configured to operate in this mode.
2. **Interrupt Frequency Moderation** - Frequency of Interrupts' generation can be trimmed to certain limit. Each mean of Interrupt Generation can be independently configured to trim

interrupts' frequency. Each MSI-X message can be independently configured to operate in this mode.

8.20.19 Completion Event Moderation

Completion Event Moderation (CEM) delays posting of Completion Event by armed CQ until multiple completions are reported on that queue. This enables to amortize overhead of completion event handler call across multiple work completions.

CEM can be enabled for each CQ individually.

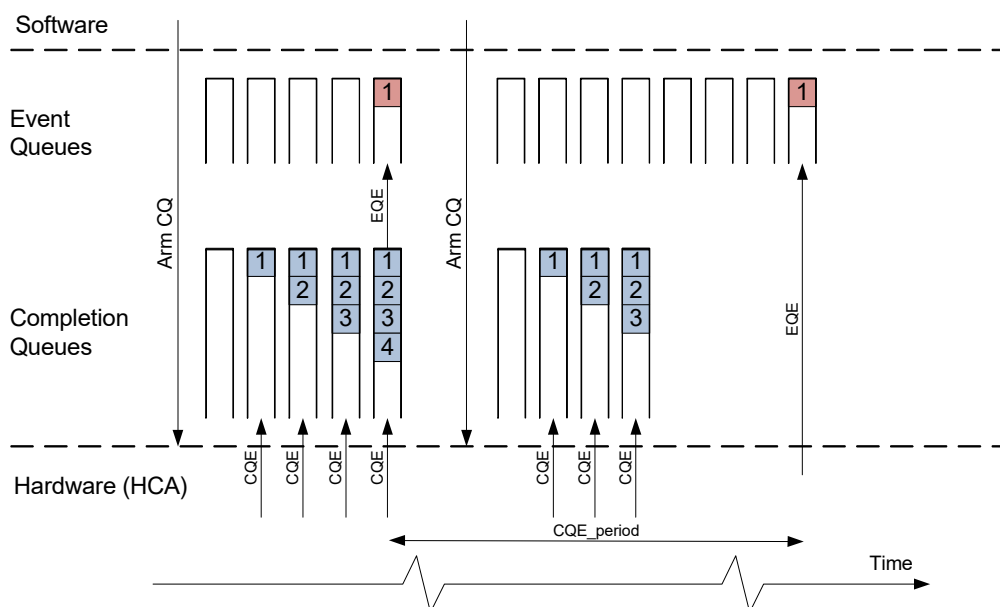
An armed CQ will generate an event when either one of the following conditions is met:

- The `cq_period` timer has expired and there is a pending event for this CQ (an event that would have otherwise been generated if it would not be subject to moderation). The `cq_period` timer restarts either upon event generation or upon completion generation, depending on the `CQ.cq_period_mode` (See Table 171, "Completion Queue Context Layout," on page 467).
- The number of completions generated since the one which triggered the last event generation reached `cq_max_count`.

Both `cq_max_count` and `cq_period` parameters are configured in CQ that is subject to CEM.

Figure 36 illustrates event generation from CQ configured to report event after 4 completions have been posted (left) or after timeout since the last completion report expired (right).

Figure 36: Completion Event Moderation



CQ is transferred to *fired* state after event has been reported.

CEM is expected to be deployed on CQs that consolidate completion reports from multiple independent and concurrently-operating WQs.

For CQs with `CQ.scqe_break_moderation_en==1`, solicited CQE (CQE.SE flag is enabled) breaks moderation and generates an immediate EQE.

Note that this feature is supported if `HCA_CAP.scqe_break_moderation` is enabled.

8.20.20 Interrupt Frequency Moderation

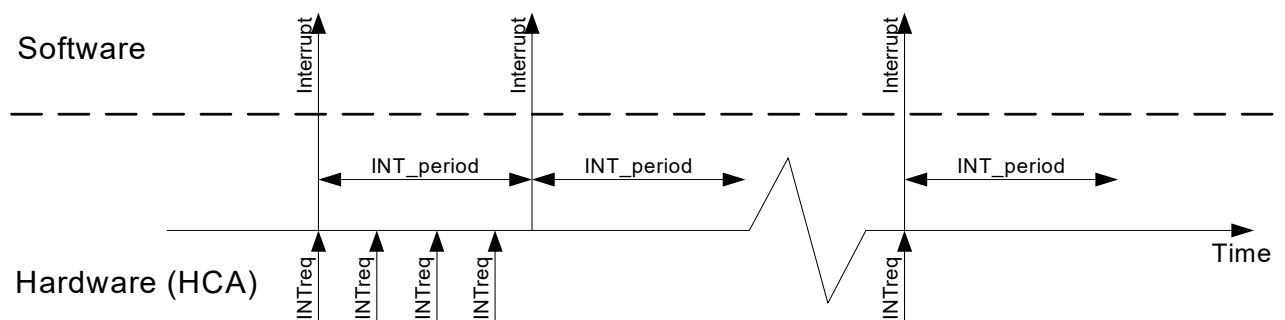
Interrupt Frequency Moderation (IFM) trims the frequency of interrupt assertion on the chip interface.

IFM is implemented on each interrupt (pin or message) and it assures that this interrupt is not asserted at a rate higher than one configured. Each interrupt message/pin can be independently configured to trim interrupt frequency; this property is configured in *Interrupt Context Table*. After interrupt request is generated on IFM-enabled interrupt, HW checks the value of interrupt-associated timer. If this timer value is zero, interrupt is generated on the chip interface, timer is re-loaded with `int_period` field of the respective entry in Interrupt Context Table and timer starts count-down. If timer value was not zero when interrupt request was generated, the interrupt generation on the chip interface will be delayed until timer is expired. Every time interrupt is fired (generated) on chip interface, the timer is re-loaded and starts count-down.

[XXX - Specify Interrupt Context Table](#)

This mechanism assures that interrupts are not generated more frequently than `int_period` time ticks apart. However, if interrupt has not been generated for a long time, the interrupt request will result in immediate generation of interrupt on chip interface. Figure 37 illustrates IFM mechanism.

Figure 37: Interrupt Frequency Moderation



Interrupt Frequency Moderation can always be deployed to assure that interrupts are not generated by the HCA at excessive rate. Interrupt moderation is configured through the `CONFIG_INT_MODERATION` command.

8.21 XRQ Context

Note: This section is preliminary and subject to change.

InfiniBand RQ (XRQ) object holding the destination for incoming packets/messages including certain offloads. XRQs are attached to the QP and report completion on their completion queue (when QP is connected with XRQ the *cqe_recv* should be reserved).

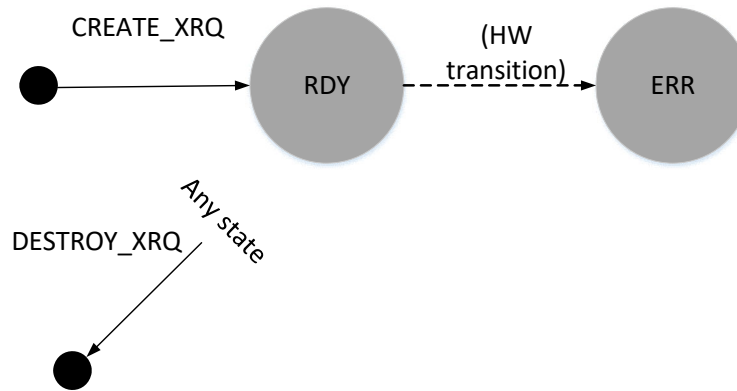
Multiple application offloads are supported by XRQ. Each offload type extends the regular functionality of the RQ to enable a specific offload. In case application offload is given, a corresponding application context should also be given when creating the XRQ.

XRQ context is created through the CREATE_XRQ command. The context can be then modified through the MODIFY_XRQ command. It can be queried through the QUERY_XRQ command. When finalized, XRQ context is destroyed through the DESTROY_XRQ command.

The number of supported XRQs is reported by *HCA_CAP.log_max_xrq*.

XRQ is subject to well defined state transitions as illustrated in the following figure.

Figure 38: XRQ State Machine



8.21.1 XRQ Context

Table 247, “XRQ Context Field Descriptions,” on page 511 describes the XRQ Context.

Table 246 - XRQ Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
state				rlky																	topology								offload				00h
								user_index																				04h					
								cqn																				08h					
																										basic_cyclic_rev_wqe	offloaded_dc_enable	0Ch					

Table 246 - XRQ Context Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																10h-1Ch
tag_matching_topology_context (See Table 248, “Tag Matching Topology Context Layout,” on page 514)																																20h-2Ch
nvme_offload_context (See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519)																																30h-6Ch
																																70h-7Ch
wq (See Table 126, “Work Queue (WQ) Format,” on page 389)																																80h-...

Table 247 - XRQ Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	state	XRQ state 0x0: READY 0x1: ERROR Reserved on CREATE_XRQ	
	27	rlky	Reserved lkey enabled. When set, the reserved lkey can be used on this XRQ.	
	11:8	topology	Topology of the XRQ. 0x0: None - No special topology. 0x1: tag_matching - Tag matching topology. valid only when HCA_CAP.tag_matching is set.	
	3:0	offload	The offload provided by the XRQ. 0x0: None - No special offload. 0x1: rndv - valid only when HCA_CAP.rndv_offload_* is set and tag_matching_topology is selected. 0x2: NVMe-oF - valid only when HCA_CAP.nvmf_target_offload is set.	

Table 247 - XRQ Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	23:0	user_index	An opaque identifier which software sets, which will be reported to the Completion Queue. 0xffffffff - user_index does not exist (e.g like cqe_version==0). Reserved when HCA_CAP.cqe_version==0. Note that in case a QP is connected to XRQ, then user_index of XRQ is reported in the responder CQE (for requester CQE will be the user_index of the QP).	
08h	23:0	cqn	CQ for XRQ to report completion on.	
0Ch	1	basic_cyclic_rcv_wqe	0: cyclic receive WQE always includes the 16 byte of ctrl (signature field). 1: cyclic receive WQE includes the 16 bytes of ctrl only when wq_signature==1. Valid only if HCA_CAP.basic_cyclic_rcv_wqe==1 and wq.wq_type==WQ_CYCLIC	
	0	offloaded_dc_enable	If set, XRQ supported offloads for dynamically transport service. SW must set the XRQ_DC_PARAMS_ENTRY using SET_XRQ_DC_PARAMS_ENTRY command, See .	
20h-2Ch	128	tag_matching_topology_context (See Table 248, “Tag Matching Topology Context Layout,” on page 514)	Tag matching topology context. Valid only when topology == tag_matching.	

Table 247 - XRQ Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
30h-6Ch	512	nvme_ofload_context (See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519)	NVMe-oF offload context. Valid only when offload == NVMF.	
80h-...	1600	wq (See Table 126, “Work Queue (WQ) Format,” on page 389)	Work Queue. Note that in case that PASs list in the WQ indicates to multiple groups of PASs (like in NVMe-oF), then each one of them must be aligned to 64 Bytes. For example, in NVMe-oF offload, the Pas in WQ contains the physical addresses of the RQ for receiving non offloaded NVMe-oF commands, followed by the physical addresses of the staging buffer. The second list must be allocated and aligned to 64 Bytes. (See Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999). Note that when WQ.wq_umem_valid==1, then WQ doesn’t contain staging buffer memory. In that case staging buffer is define by nvme_offload_context.staging_buffer_umem_id/staging_buffer_umem_size/staging_buffer_umem_offset.	

8.21.2 XRQ With Tag Matching Topology

Choosing tag matching topology for XRQ enables it to handle tag matching operation.

When tag matching topology is chosen, the regular XRQ is extended with new elements such as matching list and phase counters. SW can manage the XRQ matching list using list operation WQEs. Tag matching offload is characterized with unique CQE generation.

Figure 39, “XRQ Tag Matching Topology” describes the XRQ with tag matching offload.

Figure 39: XRQ Tag Matching Topology

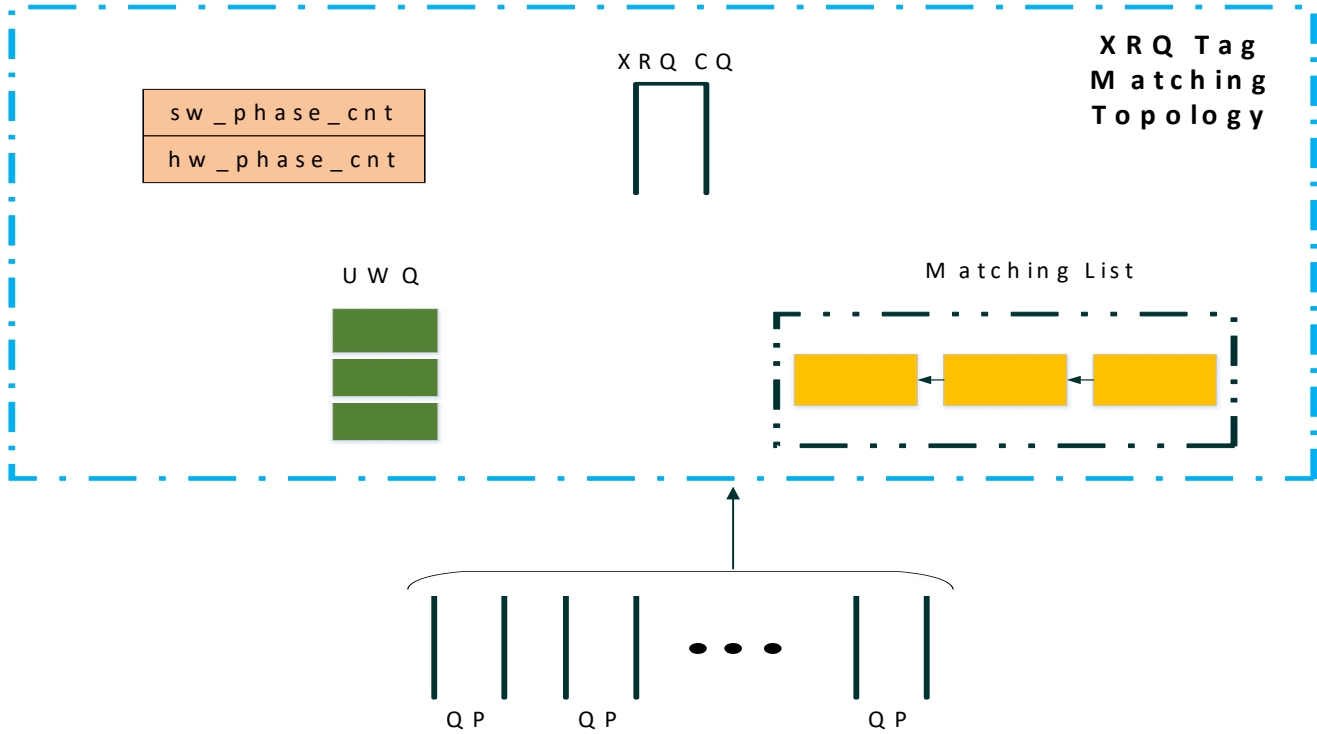


Table 248, “Tag Matching Topology Context Layout,” on page 514 describes the tag matching topology context.

Table 248 - Tag Matching Topology Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
log_matching_list_sz																append_next_index												00h				
sw_phase_cnt																hw_phase_cnt												04h				
																														08h-0Ch		

Table 249 - Tag Matching Topology Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	log_matching_list_sz	Log (Base 2) of the maximal number of matching entries in matching list. Value must be lower equal to <code>HCA_CAP.log_tag_matching_list_sz</code> .	
	15:0	append_next_index	The index to which the next append operation will be inserted. Only in QUERY command.	
04h	31:16	sw_phase_cnt	Value of <code>sw_phase_cnt</code> . Only in QUERY command.	
	15:0	hw_phase_cnt	Value of <code>hw_phase_cnt</code> . Only in QUERY command.	

8.21.2.1 Matching List

Head of SW matching list. Each element in this list is called a matching entry. Each matching entry contains a 64b tag and a pointer to a buffer

Upon successful tag matching operation, the matching entry is removed from the matching list and the message (i.e. payload after tag matching headers) is scattered to the matching entry buffer. The TMH will be scattered to a tag consumed CQE.

Upon unsuccessful tag matching operation, the message and the tag matching headers are scattered to the next valid WQE from the UWQ. SW appends or removes tags using list operation WQE and adds a reference to send WQE format. The maximum number of tags in XRQ is reported through `HAC_CAP.log_matching_list_sz`.

Only messages with operation that carries tag will search the matching list.

8.21.2.2 UWQ - Unexpected WQ

UWQ contains WQEs to which unexpected messages and not tagged will be scattered.

UWQ size and topology (i.e. link list, cyclic etc) are set according to WQ parameters in `CREATE_XRQ` command.

Note: Implementation demands that UWQ will have at least 32 pre-posted WQEs. If WQE counter drops below this number, the device will drop incoming message and send RNR.

8.21.2.3 SW and HW Phase Counters

`sw_phase_cnt` and `hw_phase_cnt` are used to satisfy tag matching ordering rules. When counters are not equal, new tag appends do not take part in the tag matching operation. When counters are equal, all tags in matching list take part in the tag matching operation.

hw_phase_cnt field is increased by 1 every time HCA passes an unexpected message to SW. *sw_phase_cnt* is updated when SW updates it using list operation WQE.

Note: In case UWQ type is striding, there will be multiple “message_arrived_unexpected” CQEs per message. However, the *hw_phase_counter* will advance only by 1.

SW should track the counters status and avoid counters wraparound to preserve the ordering rules of tag matching.

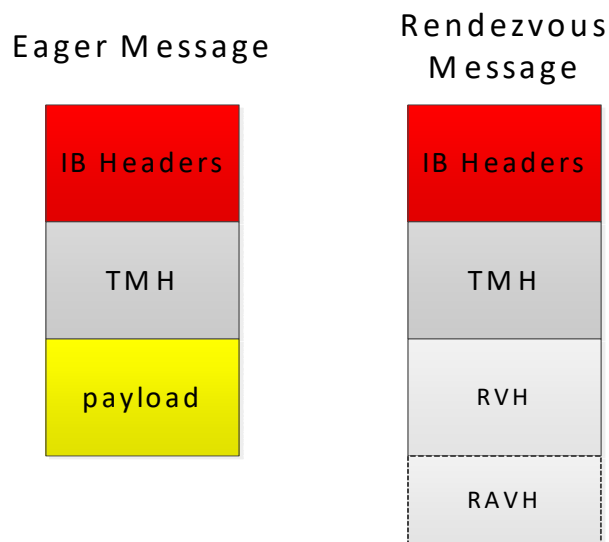
8.21.2.4 Tag Matching Headers

Tag matching headers are in Big Endian and added for messages that require tag matching.

Tag matching header (TMH) is always mandatory.

When TMH.protocol=*rendezvous*, SW should not add any other headers/payload except the RVH and RAVH. Rendezvous header (RVH) is mandatory. Response AV header (RAVH) is optional. It specifies the address vector for the rendezvous completion operation performed by rendezvous QP, hence needed only when the remote rendezvous QP (owned by the device) does not have explicit fixed connection. For example, RAVH is needed when remote rendezvous QP transport service is DCI but not needed when remote rendezvous QP transport service is RC.

Figure 40: Tag Matching Headers



SW can choose the protocol of point-to-point communication from the below list:

- Rendezvous - SW uses this protocol for RTS (Ready to send) messages. The device completes the message handling by initiating RDMA_READ operation to obtain the remote data. In case rendezvous opcode is used, SW must add the RVH to payload.
- No Tag - in no tag protocol, the tag value in TMH is ignored. XRQ does not attempt to perform tag matching on those messages and scatter them to a WQE from UWQ. In addition, neither HW nor SW will advance *hw_phase_cnt* or local *sw_phase_cnt* values.

- Eager - in Eager protocol, the target XRQ performs tag matching on the message. In case matching entry was found, the message will be scattered to the matching entry buffer.

Table 250 - TMH - Tag Matching Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
operation							has_tag																					00h				
application_context																											004h					
tag																											08h-0Ch					

Table 251 - TMH - Tag Matching Header Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:25	operation	Message operation: when has_tag==1: 0x0: rendezvous 0x1: eager When has_tag==0: 0x0: reserved 0x1: FIN	
	24	has_tag	If set, then message carries a tag to be handled by the device.	
04h	31:0	application_context	An opaque for application inner usage. Not valid in “no_tag” protocol.	
08h-0Ch	64	tag	64b tag for tag matching operation. Not valid in “no_tag” protocol.	

Table 252 - RVH - Rendezvous Header Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	va	Virtual address of remote data.	
08h	31:0	r_key	Remote memory key for the RDMA transaction.	

Table 252 - RVH - Rendezvous Header Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
0Ch	31:0	dma_length	The RDMA transaction length.	

Table 253 - RVH - Rendezvous Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
va																																00h-04h
r_key																																08h
dma_length																																0Ch

Table 254 - RAVH - Response AV Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
sl_or_eth_prio								DCT																									00h
																																	004h
dc_access_key																																	08h-0Ch

Table 255 - RAVH - Response AV Header Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	sl_or_eth_prio	For IB - sl to be used for rendezvous completion. For RoCE: eth_prio[27:25] - ethernet priority, eth_prio[24] = dei/cfi. To be used for rendezvous completion.	
	23:0	DCT	DCT as target for rendezvous completion - in case remote QP is associated with DC rendezvous QP. The LID for rendezvous completion will be according to rndv control slid.	
08h-0Ch	64	dc_access_key	DC access key to be used in rendezvous completion packet. valid only when addressing XRQ with DC transport	

8.21.3 NVMe Express over Fabric (NVMe-oF) Target Offload Context

The device can be used to enable support of NVMe over fabric target offload by XRQ.

The NVMe-oF target offload context should be passed to the device when creating XRQ using CREATE_XRQ command.

Table 256 - NVMe-oF Target Offload Context Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nvmf_offload_type										nvmf_stats_report_back_en	log_max_name-space																				00h	
cmd_unknown_namespace_cnt																																04h
ioccsz																																08h
icdoff												log_max_io_-size							log_rmp_size								nvme_memory_log_page_size				0Ch	
			log_max_meta-data_size							stag-ing_buffer_log_page_size				staging_buffer_number_of_pages																10h		
staging_buffer_page_offset																																14h
error_report_filter																nvme_queue_size																18h
never mask more bit	lossless error report							error_report_qpn																								1Ch
staging_buffer_umem_id																																20h
staging_buffer_umem_size																																24h
staging_buffer_umem_offset																																28h-2Ch

Table 256 - NVMe-oF Target Offload Context Header Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
<u>ioresz</u>																																30h				
				<u>log_report_in_flight_step_size</u>				<u>report_in_flight_level0</u>																								34h				
				<u>report_back_location</u>												<u>report_pending_cmds_level0</u>																38h				
																				<u>num_bits_for_in_flight_data</u>								<u>log_report_pending_cmds_step_size</u>								3Ch

Table 257 - NVMe-oF Target Offload Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	<u>nvm-f_offload_type</u>	The supported NVMe over Fabrics offloaded operations: 0x1: NVMe_WRITE - NVMe WRITE will be offloaded. Other operations will be transferred to RQ for application level processing. Supported only when <i>nvme_write_offload</i> in HCA_CAP is set. 0x2: NVMe_READ - NVMe READ will be offloaded. Other operations will be transferred to RQ for application level processing. Supported only when <i>nvme_read_offload</i> in HCA_CAP is set. 0x3: NVMe_READ_WRITE - NVMe READ, WRITE will be offloaded. Other operations will be transferred to RQ for application level processing. Supported only when <i>nvme_read_write_offload</i> in HCA_CAP is set. 0x4: NVMe_READ_WRITE_FLUSH - NVMe READ, WRITE and FLUSH will be offloaded. Other operations will be transferred to RQ for application level processing. Supported only when <i>nvme_read_write_flush_offload</i> in HCA_CAP is set.	
	21	<u>nvm-f_stats_report_back_en</u>	Enable the "report back" feature for this XRQ, which means that NVMe-oF Completion Capsules sent to the initiator include internal XRQ statistics. Supported when HCA_CAP.report_back_supported==1. Requires setting <i>ioresz</i> > 1. [Internal] This is a non-standard feature, existing initiators may not handle these completions correctly.	
	20:16	<u>log_max_namespace</u>	Log (base 2) of the maximum number of namespaces that can be attached to this XRQ.	

Table 257 - NVMe-oF Target Offload Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	31:0	cmd_unknown_namespace_cnt	Number of commands with unknown (non existing) name space. Supported only when HCA_CAP.cmd_unknown_namespace_cnt==1.	
08h	31:0	ioccsz	I/O Queue Command Capsule Supported Size (IOCCSZ). This field defines the I/O command capsule size in 16 byte units. Must be between min_ioccsz and max_ioccsz in HCA_CAP. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.	
0Ch	31:24	icdoff	This field defines the offset where data starts within a capsule. This value is applicable to I/O Queues only (the Admin Queue shall use a value of 0h). The value is specified in 16 byte units. The offset is from the end of the Submission Queue Entry within the command capsule (starting at 64 bytes in the command capsule). The minimum value is 0 and the maximum value is reported in HCA_CAP.max_icdoff.	
	20:16	log_max_io_size	Log (base 2) of the maximum supported NVMe I/O size in granularity of bytes. Must be <= HCA_CAP.NVMF_CAP.log_max_io_size. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.	
	12:8	log_rmp_size	Log (base 2) of the number of WQEs in the RMP. Valid only upon Query.	
	3:0	nvme_memory_log_page_size	Log (base 2) of NVMe memory page size in granularity of 4KB. Note that staging_buffer_log_page_size must be >= nvme_memory_log_page_size.	
10h	28:24	log_max_metadata_size	Log(base 2) of the maximum supported metadata size per nvme io transaction. 0 means no MetaData. Must be <= HCA_CAP.log_max_metadata_size	
10h	20:16	staging_buffer_log_page_size	Log (base 2) of page size of staging buffer in units of 4Kbyte. Reserved when XRQ.WQ.wq_umem_valid==1.	
	15:0	staging_buffer_number_of_pages	Number of pages of staging buffer. Staging buffer size in Bytes is staging_buffer_number_of_pages*2^staging_buffer_log_page_size. Staging buffer size in bytes must be between 2^(log_min_staging_buffer_pages+nvme_memory_log_page_size) and 2^(log_max_staging_buffer_pages+nvme_memory_log_page_size). Reserved when XRQ.WQ.wq_umem_valid==1.	

Table 257 - NVMe-oF Target Offload Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
14h	31:0	stag- ing_buff er_page_ offset	Page offset in granularity of $2^{\log\_memory\_page\_size}$. Reserved when $XRQ.WQ.wq\_umem\_valid==1$.	
18h	31:16	error_re- port_fil- ter	A bitmask indicates which errors to report using <code>error_report_qpn</code> . See <code>HCA_CAP.nvmf_error_filters</code> for the supported filters and enumeration.	
	15:0	nvme_ queue_ size	Size of each of the NVMe sq and NVMe cq of all NVMe backend controller attached to this XRQ. Must be in range of <code>HCA_CAP.log_min_queue_size</code> and <code>HCA_CAP.log_max_queue_size</code> . 0 means 2^{16} . See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780.	
1Ch	31	loss- less_er- ror_repor t	If set to 0, device may drop errors if the device side send queue is full. If set to 1, the device will not lose an error, and if there are no send credits or the errors QP is in ERROR state, the device delays sending the response back to the host until it is able to post. Supported only when $HCA\_CAP.lossless\_error\_report==1$.	
	30	nev- er_mask _more_bi t	Relevant when $lossless\_error\_report=0$ or when $error\_report\_qpn==0$. When set to 0 and error log message can't be posted on the errors QP, the device will mask (clear) the More bit in the CQE sent to the host. This is in order to comply with the nvme spec. If set to 1, no such masking will take place. Supported only when $HCA\_CAP.nvmf\_error\_reporting==1$	
	23:0	error_re- port_qpn	A QP number that is used for reporting errors from the NVMe target offload engine to software. When 0, feature is disabled and no errors are reported. Before creating the XRQ, the QP must be a reliable connected (RC) QP in RTS state. Upon successful completion of the command, the QP is owned by the device, and software should not try to post anything on it. On DESTROY_XRQ the QP returns to software ownership. The Error Message format is described in See Table 258, “NVMe Error Message Layout,” on page 524 . Supported only when $HCA\_CAP.nvmf\_error\_reporting==1$	
20h	31:0	stag- ing_buff er_umem_ id	A UMEM object ID for staging buffer for this subsystem. Valid when $XRQ.WQ.wq\_umem\_valid==1$.	
24h	31:0	stag- ing_buff er_umem_ size	Size to be used for the staging buffer. Valid when $XRQ.WQ.wq\_umem\_valid==1$.	

Table 257 - NVMe-oF Target Offload Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
28h-2Ch	64	stag-ing_buff-er_umem_offset	Offset in the provided UMEM where the staging buffer starts. The resulting address (of this offset and the UMEM creation offset) should be page aligned. Valid when XRQ.WQ.wq_umem_valid==1.	
30h	31:0	iorcsz	<u>The CC size, in units of 16B.</u> <u>A value of 0 is the same as the value of 1 which means the minimum required CC size of 16B.</u> <u>Allowed values are between min_iorcsz and max_iorcsz in HCA_CAP.</u> <u>The extra space above 16B will be actually sent on the wire only if other features that make use of it are enabled (as report_back_en).</u>	
34h	28:24	log_re-port_in-flight_step_size	<u>Each report level is step_size, up to 16 levels.</u> <u>Above that a value of 15 will be reported.</u> <u>Log value.</u>	
	23:0	report_in-flight_level0	<u>The minimal value of in-flight data to consider for the first reporting level.</u> <u>If in-flight data is less than this, the reported in-flight data will be 0.</u> <u>In units of 128KB.</u>	
38h	27:16	report_back_location	<u>The offset within the report back space that this XRQ should report to, in bytes. Software must make sure not to assign the same report_back_location to more than one XRQ or reporting will be undefined. Setting a location larger than the possible space according to IORCSZ is invalid. The actual response capsule size will be the minimum required to accommodate the highest report_back_location.</u>	
	15:0	report_pending_cmds_level0	<u>The minimal value of pending commands to consider for the first reporting level. If pending commands is less than this, the reported pending commands will be 0.</u>	
3Ch	11:8	num_bits_for_in-flight_data	<u>How many bits of the 8 bits report byte to be used for in flight data reporting. The rest will be used for pending commands reporting. Possible values are between 0-8. 0 means no in flight data reporting. 8 means no pending commands reporting.</u>	
	4:0	log_report_pending_cmds_step_size	<u>Each report level is step_size, up to 16 levels.</u> <u>Above that a value of 15 will be reported.</u> <u>Log value.</u>	

The Table below describes the NVMF Error Message that is reported by the device to SW using error_report_qpn. The message size is reported in HCA_CAP.nvmf_error_message_size.

Table 258 - NVMF Error Message Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
error_type								xrqn																								00h
								qpn																								004h
smac (See Table 33, “MAC Address Layout,” on page 293)																															08h-0Ch	
sgid																															10h-1Ch	
cid															status																20h	
param_loc																															24h	
lba																															28h-2Ch	
nsid																															30h	
backend_id																															34h	
backend_cid																															38h	
																															3Ch	
backend_cqe																															40h-4Ch	
																															50h-7Ch	

Table 259 - NVMF Error Message Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	error_type	Error Type. 0x0: Offload_Engine_ERROR 0x1: NVME_ERROR_COMPLETION 0x2: NVME_GOOD_COMPLETION - TBD Oren? what does mean.	
	23:0	xrqn	XRQ number	
04h	23:0	qpn	QP number on which the original CC arrived.	

Table 259 - NVMe Error Message Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
08h-0Ch	64	smac (See Table 33, “MAC Address Layout,” on page 293)	Source Mac Address. See Table 33, “MAC Address Layout,” on page 293	
10h-1Ch	128	sgid	Source GID. 0 means GID not present.	
20h	31:16	cid	The CID of the original CC	
	15:0	status	An applicable NVMe status code (by NVMe spec)	
24h	31:16	param_loc	Parameter Error Location as defined in NVMe spec	
28h-2Ch	64	lba	The LBA where error happened	
30h	31:0	nsid	The Namespace ID of the original CC	
34h	31:0	back-end_id	The backend used to submit the NVMe command (relevant for types 0x2, 0x3)	
38h	31:16	back-end_cid	The CID used for the submitted NVMe command (relevant for types 0x2, 0x3)	
40h-4Ch	128	back-end_cqe	The original NVMe completion (relevant for types 0x2, 0x3)	

8.22 Transport Static Params Context

This context allows SW to update the device with static parameters for TLS offload, such as the TLS connection crypto key, TLS version, etc.

The context is created automatically when creating TIS/TIR with TLS offload.

Updating this context is done by UMR WQEs where:

- wqe op_mode = TLS_TIS_STATIC_PARAMS/TLS_TIR_STATIC_PARAMS
- TIR/TIS number in general_ctrl_id[31:8]

Table 260 - Transport Static Params Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
con- st_2		tls_version				con- st_1																				encryption- standard- c_type		00h						
																																04h		
initial_record_number																																08h-0Ch		
resync_tcp_sn																																10h		
gcm_iv																																14h		
implicit_iv																																18h-1Ch		
								dek_index																								20h		
																				ti		zero copy en		ddgst offload en		hdgst offload en		ddgst en		hddgst en		pda		24h
nvme_resync_tcp_sn																																28h		
																																2Ch-3Ch		

Table 261 - Transport Static Params Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:30	const_2	Must be set to 2.	
	29:26	tls_version	TLS version. 0x2: TLS_1_2 0x3: TLS_1_3	
	25:24	const_1	Must be set to 1.	
	3:0	acc_type	0x1: TLS 0x2: NVMETCP 0x3: NVMETCP_with_TLS	

Table 261 - Transport Static Params Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
08h-0Ch	64	initial_record_number	Record number initial value.	
10h	31:0	resync_tcp_sn	Used for resync flow. See TBD (resync section).	
14h	31:0	gcm_iv	GCM IV.	
18h-1Ch	64	implicit_iv	Implicit IV	
20h	23:0	dek_index	DEK index	
24h	10	ti	0x0: INITIATOR 0x1: TARGET Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	9	zero_copy_en	If set, device performs zero copy offload on received data. Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	8	ddgst_offload_en	Device preforms digest offload on the PDU data (SX and RX) Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	7	hdgst_offload_en	Device preforms digest offload on the PDU header (SX and RX) Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	6	ddgst_en	If set, data digest is used for the connection. If cleared, data digest is not used for the connection Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	5	hddgst_en	If set, header digest is used for the connection. If cleared, header digest is not used for the connection. Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
	4:0	pda	0-31 (0-4B align, 1-8B align, 2-12B...) Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	
28h	31:0	nvme_resync_tcp_sn	Used for resync; SW copies the resync sn read from PSV, HW will return to offload only when they match. Valid only when acc_type==NVMETCP/NVMETCP_with_TLS.	

8.23 TLS PROGRESS Params Context

This context is shared between the device and software. Software must initialize it when starting a TLS connection, the device updates the TLS connection volatile states. SW write to this context during TLS offload is forbidden, but it's possible to read the context and follow the device state.

In addition, the context holds the states of 2 state machines:

For resync scenario (see [Section 4.9.3.1, “Resync Flow,” on page 219](#)) the context will hold the TCP-SN of the packet where resync took place. This information is crucial for returning to offload state. SW must specify this exact info in the Transport Static Params Context.resync-tcp-sn field for the device to return to offload state.

Record_tracker: tracks the TLS records. At an event of packet loss this state moves from tracking to searching. When the device has detected a record it will move to tracking, this is not a sufficient condition for offloading.

Auth_state: the device offload state, based on the tracking state and GCM authentication success. The device will move to offloading state only if the record_tracker is in tracking state, and a full record has been authenticated, and the TNV.resync-tcp-sn field is equivalent to the HW_resync_tcp_sn in this context.

The context is created automatically when creating TIS/TIR with TLS offload.

Update/Query this context is done by SET_PSV/GET_PSV WQEs where:

- wqe op_mode = TLS_TIS_STATIC_PARAMS/TLS_TIR_STATIC_PARAMS
- TIR/TIS number instead of psv_index

For more details, refer to TBD.

Table 262 - TLS Progress Params Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
next_record_tcp_sn																																00h	
hw_resync_tcp_sn																																04h	
recor d_tra cker_ state	auth_ state							hw_offset_record_number																									08h

Table 263 - TLS Progress Params Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	next_re- cord_tcp_sn	Expected TCP-SN of next record	
04h	31:0	hw_resyn- c_tcp_sn	TCP-SN of the record where resync began (reset record counter)	

Table 263 - TLS Progress Params Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
08h	31:30	record_tracker_state	Record tracker state 0x0: START 0x1: TRACKING 0x2: SEARCHING	
	29:28	auth_state	Authentication state 0x0: NO_OFFLOAD 0x1: OFFLOAD 0x2: AUTHENTICATION	
	23:0	hw_offset_record_number	Offset of record counter relative to Transport_Static_Params.initial_record_number, incremented by HW. See Section 4.9.3.1, "Resync Flow," on page 219	

8.24 NVMeoTCP Progress Params Context

For more details, refer to TBD.

Table 264 - NVMeoTCP Progress Params Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
valid									pd																												00h
next_pdu_tcp_sn																																04h					
hw_resync_tcp_sn																																08h					
pdu_tracker_state	off-loading_state															cccid_ttag																0Ch					

Table 265 - NVMeoTCP Progress Params Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	valid	Context is valid	
	23:0	pd	Protection domain	
04h	31:0	next_pdu_tcp_sn	Expected TCP-SN of next pdu	
08h	31:0	hw_resync_tcp_sn	TCP-SN of the pdu where began resync (reset record counter)	
0Ch	31:30	pdu_tracker_state	0x0: START 0x1: SEARCHING 0x2: TRACKING	
	29:28	offloading_state	0x0: NO_OFFLOAD 0x1: OFFLOAD_ENABLED	
	15:0	cccid_ttag	CCCID or TTAG (according to IT field in NVMeoTCP Static Params)	

8.25 FPGA QP

This section describes the FPGA QP which allows traffic communication with the FPGA.

FPGA QP is created through the `FPGA_CREATE_QP` command. The context can be then modified according to the state machine through the `MODIFY_SQ` command. It can be queried through the `FPGA_QUERY_QP` command. When finalized, FPGA QP context is destroyed through the `FPGA_DESTROY_QP` command. See Section 29.20, “FPGA QP Commands,” on page 2063.

The supported types of FPGA QPs and the number of supported QPs is reported via the `FPGA_CAP` register through the *qp_type* and *max_num_qps* parameters. See Table 891, “FPGA_CAP Register Layout,” on page 1103.

FPGA context is subject to well defined state transitions as illustrated in Figure 26.

Figure 41: *FPGA QP States*



FPGA QP context format is shown in Table 138.

Table 266 - *FPGA QP Context Format*

3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	2	1	0	9	8	7	6	5	4	3	2	1	0												qp_type	00h
state																														04h					
			st																				traffic_class				04h								
ether_type														prio		dei	vid						08h												
																												0Ch							
				next_rcv_psn																								10h							
				next_send_psn																								14h							

Table 266 - FPGA QP Context Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										18h	
																pkey																18h
								remote_qpn																								1Ch
																		rnr_re-try								retry_c-ount				20h		
																																24h
																remote_mac[47:32]																28h
remote_mac[31:0]																																2Ch
remote_ip																																30h-3Ch
																																40h-44h
																fpga_mac[47:32]																48h
fpga_mac[31:0]																																4Ch
fpga_ip																																50h-5Ch

FPGA QP Context fields descriptions are shown in Table 139.

Table 267 - FPGA QP Context Fields

Offset	Bits	Name	Description
00h	31:28	state	FPGA QP state 0x0: INIT 0x1: ACTIVE 0x2: ERROR
	0	qp_type	FPGA QP type to be created 0x0: SHELL_QP - supported only when reported in FPGA_CAP.shell_caps.qp_type. 0x1: SANDBOX_QP - supported only when reported in FPGA_CAP.shell_caps.qp_type. See Table 893, “Shell_Caps Layout,” on page 1107
04h	27:24	st	Transport service type 0x0: RC - supported only when FPGA_CAP.shell_caps.rc==1. See Table 893, “Shell_Caps Layout,” on page 1107
	7:0	traffic_class	Traffic class
08h	31:16	ether_type	VLAN ethertype. FPGA expects to receive traffic on this ethertype and it responds to it.
	15:13	prio	VLAN priority. FPGA expects to receive traffic on this prio and it responds to it.
	12	dei	dei in VLAN header. FPGA expects to receive traffic on this dei and it responds to it.
	11:0	vid	VLAN ID. FPGA expects to receive traffic on this vid and it responds to it.
10h	23:0	next_rcv_psn	Next PSN (expected) to be received (from FPGA perspective)
14h	23:0	next_send_psn	Next PSN to be sent (from FPGA perspective)
18h	15:0	pkey	Partition key
1Ch	23:0	remote_qpn	QP number of the remote side
20h	10:8	rn timer	RNR NAK retry limit prior to transition to error state
	2:0	retry_count	Transport timeout retry count
28h	15:0	remote_mac[47:32]	MSB of remote MAC address FPGA expects to receive traffic from this MAC and responds to it.
2Ch	31:0	remote_mac[31:0]	LSB of remote MAC address. FPGA expects to receive traffic from this MAC and responds to it.

Table 267 - FPGA QP Context Fields

Offset	Bits	Name	Description
30h-3Ch	128	remote_ip	Remote IP address. FPGA expects to receive traffic from this IP and responds to it.
48h	15:0	fpga_mac[47:32]	MSB of FPGA QP MAC address. Valid only on FPGA_QUERY_QP and output of FPGA_CREATE_QP. SW should use this address to communicate with the created QP.
4Ch	31:0	fpga_mac[31:0]	LSB of FPGA QP MAC address. Valid only on FPGA_QUERY_QP and output of FPGA_CREATE_QP. SW should use this address to communicate with the created QP.
50h-5Ch	128	fpga_ip	IP address of FPGA QP. Valid only on FPGA_QUERY_QP and output of FPGA_CREATE_QP. SW should use this address to communicate with the created QP.

To modify an FPGA QP parameter or perform state transition, software provides a bitmask of the various fields that are being changed. Table 140 summarizes the bitmasks and the supported state transitions where bits can be set.

Table 268 - FPGA_CREATE_QP and FPGA_MODIFY_QP Bitmask

Bit	Field	FPGA_CREATE_QP	INIT2ACTIVE	ACTIVE2INIT	ERR2ACTIVE
none	state	R	R	R	R
none	qp_type	R			
none	st	R			
none	traffic_class	R			
none	ether_type	R			
none	prio	R			
none	dei	R			
none	vid	R			
none	next_rcv_psn	R			
none	next_send_psn	R			
none	pkey	R			

Table 268 - FPGA_CREATE_QP and FPGA_MODIFY_QP Bitmask

Bit	Field	FPGA_CREATE_QP	INITIATIVE	ACTIVE2INIT	ERRACTIVE
none	remote_qpn	R			
none	rnr_retry	R			
none	retry_count	R			
none	remote_mac	R			
none	remote_ip	R			
none	fpga_mac	R			
none	fpga_ip	R			

Legend

- R - Required parameter
- O - Optional parameter

8.25.1 FPGA QP States Summary

Table 269 summarizes the FPGA QP states.

Table 269 - FPGA QP States Summary

State	Description
INIT	QP cannot send or receive traffic.
ACTIVE	Send and receive traffic processed normally.
ERROR	QP cannot send or receive traffic. It is SW responsibility to return the QP to INIT state and then to activate it again.

8.26 Transmit Scheduler

The Transmit Scheduler is responsible for selecting the next entity to serve, on the transmit path. By configuring the Transmit Scheduler, QoS policies may be enforced, between the competing entities. For every scheduling cycle, the selected entity will be allowed to transmit a certain amount of data.

8.26.1 Scheduling Arbiter

The basic scheduling building block, is the Transmit Scheduling Arbiter (TSAR). The arbitration defines the QoS policy between the agents connected to the TSAR. The winner of the arbitration is the agent to be serviced. The functional diagram of the TSAR is shown in Figure 42 on page 536.

8.26.1.1 Bandwidth Allocation

The TSAR implements a Deficit Weighted Round Robin (DWRR) between the agents. Each agent attached to the TSAR is assigned with a Weight. An agent is awarded with transmission tokens according to its Weight, and charged with transmission Tokens according to the amount of data it has transmitted. Effectively, the Weight (relative to the other agents' Weight) defines the percentage of the BW an agent will receive, assuming it has enough data to sustain this BW.

This arbitration scheme is work-preserving, meaning that an agent not using the entire BW it was allocated, hands over the excess BW, to be redistributed among the other agents. Each agent will receive additional BW according to its Weight.

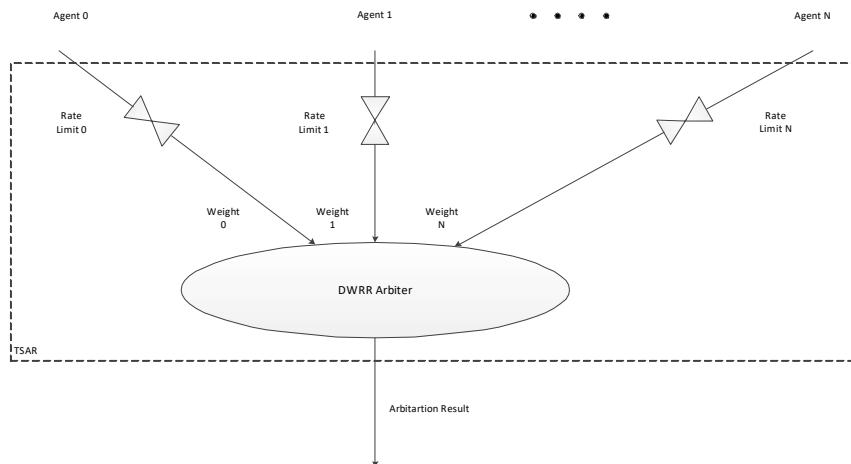
TBD: accuracy

8.26.1.2 Rate Limit

Each agent attached to the TSAR is optionally assigned with a Rate Limit. TSAR will not allow scheduling for an agent exceeding its defined Rate Limit

TBD: accuracy

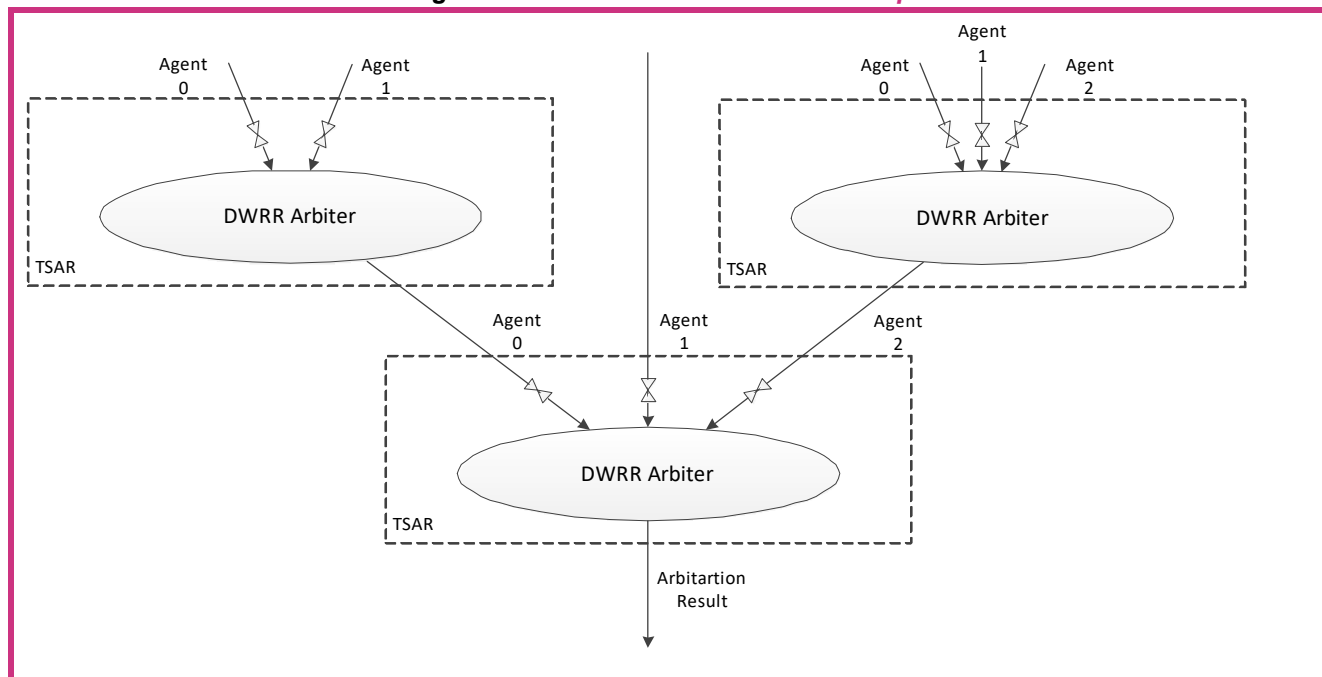
Figure 42: TSAR structure



8.26.2 Multi-level scheduler

The Transmit Scheduler can be built out of one or more TSARs. The arbitration result of each TSAR is connected to an agent of another TSAR. The lowest level TSAR is referred to as the root TSAR, and it is not connected to another TSAR. An example of a multi-level scheduler is shown in Figure 43 on page 537

Figure 43: Multi-Level Scheduler Example



8.26.3 Special TSAR Types

8.26.3.1 ETS TSAR

A TSAR may represent a DWRR between TCs. In most cases, it is desirable that this TSAR will match the ETS configurations, otherwise the resulting arbitration will not comply with either definition. Since ETS is managed through a different interface (TBD) and may also be managed internally by the device, a TSAR representing TC arbitration should be marked as ETS TSAR.

~~ETS TSAR agents cannot set/modify their QoS attributes. `bw_share` and `max_average_bw` should be initially set to 0x0~~

A TSAR may represent a DWRR between TCs. In most cases, it is desirable that this TSAR will match the ETS configurations, otherwise the resulting arbitration will not comply with either definition. Since ETS is managed through QETCR register and may also be managed internally by the device, a TSAR representing TC arbitration should be marked as ETS TSAR. The following properties apply to ETS arbiter agents:

- each agent should specify which TC it represents when attached to the ETS arbiter. only a single agent should represent each TC.
- `bw_share` and `max_average_bw` cannot be set/modified for these agents. These parameters are reserved when the agents are attached.
- When queried, `bw_share` and `max_average_bw` reflect the values actually used for ETS arbitration (instead of the initial reserved values).

8.26.3.2 Round Robin TSAR

A round robin arbitration scheme is achievable through equal weight setting of the DWRR TSAR. However, the Round Robin TSAR is a light-weight arbiter, mostly used to “aggregate” several agents as a single arbitration entity. For example, multiple VFs may represent a single VM. The VM has a set of QoS attributes (Maximal rate, BW allocation) while no VF has any precedence over the other.

Round Robin TSAR agents cannot modify their QoS attributes. `bw_share` and `max_average_bw` should be initially set to 0x0.

8.26.4 TSAR Configuration

8.26.4.1 TSAR Management

TSARs are created using “CREATE_SCHEDULING_ELEMENT - Create a Transmit Scheduling element” command. The first TSAR created is the root TSAR. Every other TSAR should specify upon creation the TSAR it is connected to (*parent_element_id* field). The location of the TSAR in the scheduling hierarchy is specified in *element_hierarchy* field. The TSAR special type is indicated in the *tsar_type* field. The command returns a Scheduling element handle *scheduling_element_id*, that is used in conjunction with the *element_hierarchy* as a unique identifier for the TSAR in other operations.

When a TSAR is no longer required, it should be destroyed using “DESTROY_SCHEDULING_ELEMENT - Destroy a Transmit Scheduling Element” command. A TSAR may be destroyed only after all of its agents were disconnected. Destroying a TSAR also disconnects it from the parent TSAR to which it is connected.

8.26.4.2 TSAR Agent Management

Transmitting entities are connected to a TSAR using the “CREATE_SCHEDULING_ELEMENT - Create a Transmit Scheduling element” command. The *element_hierarchy* and *parent_element_id* fields identify the TSAR to which the entity will be connected. The transmitting entity is identified by *element_parameters*. The parameters depend on the type of transmitting entity specified in *element_type*. The command returns the element handle (*scheduling_element_id* field) for reference in other commands.

QoS attributes are modified using the “MODIFY_SCHEDULING_ELEMENT - Modify a Transmit Scheduling Element Attributes” commands. The *element_hierarchy* and *scheduling_element_id* fields identify the Scheduling element. The *modify_bitmask* controls which QoS attributes will be modified by the command, and which will remain unmodified. The supported QoS attributes for an element are: BW allocation and rate limit. They are specified in *bw_share* and *max_average_bw* fields respectively. The *bw_share* must not exceed *tsar_bw_share_limit* field in *max_tsar_bw_share*.

The current QoS attributes of an element can be queried using the “QUERY_SCHEDULING_ELEMENT - Query a Transmit Scheduling Element”.

When a transmitting entity is no longer required or moves to another location in the scheduler structure, it should be detached from the current TSAR agent using “DESTROY_SCHEDUL-

ING_ELEMENT - Destroy a Transmit Scheduling Element” command. The *element_hierarchy* and *scheduling_element_id* fields identify the entity.

Transmitting entities not defined as a Scheduling element will have undefined QoS attributes. When building a scheduler, all relevant transmitting entities must be connected, to achieve a correct behavior.

8.27 General Object Types

Note: This section is preliminary and subject to change.

This section describes the general object types supported by the device, and that can be created and processed using the general object commands (See Section 29.23, “General Object Commands,” on page 2079).

Support for each object is reported in HCA_CAP.general_obj_types (See Table 1571, “HCA Capabilities Layout,” on page 1671).

8.27.1 UCTX - User Context

The device supports associating objects with a user context and enforcing per-user isolation and privileges. This capability also applies to PRM objects not conforming to the generic object commands format, i.e objects that are created by non-generic object commands, e.g TIR. These objects may be associated with a user context too. The device will enforce user-isolation and permissions for such objects as well.

To achieve that, SW needs to:

- Create a user context, providing the user privilege level, and in turn get a UID (user ID) from the device.
- Use this UID when creating a context/object.

User context is created using CREATE_UCTX command providing the UCTX Object. The device returns an UID as obj_id. That user ID is then applied to all commands of that user by setting the UID field, which exists in the input mailbox of all commands.

uid=0 is the default user ID. The CREATE_UCTX command will never return uid=0. The device will not enforce any user isolation/permissions for commands tagged with uid=0.

The device output mailbox of a QUERY_HCA_CAP command tagged with uid!=0 is affected by the privileges set for the specified user upon UCTX creation (in the CREATE UCTX privilege field).

[\[TODO Achiad\]: define the detailed permissions using the VF permissions table](#)

In addition, the following commands are not allowed with uid!=0:

- ENABLE_HCA
- SET_ISSI
- QUERY_PAGES

- `MANAGE_PAGES`
- `SET_HCA_CAP`
- `INIT_HCA`
- `SET_DRIVER_VERSION`
- `CREATE/DESTROY_EQ`
- `TEARDOWN_HCA`
- `DISABLE_HCA`

[\[TODO Achiad\]: complete the list](#)

8.27.2 UMEM - User Memory Context

Objects that use PAS list to refer to host memory directly may alternatively refer to host memory indirectly by setting the `umem_valid` bit in the command inbox and providing `umem_id+umem_offset` instead of the PAS list (the PAS list value would be N/A when `umem_valid=1`).

The UMEM context is created using `CREATEUMEM` command providing UMEM OBJECT that identifies the physical memory. The device returns an `umem_id` as `obj_id`.

After obtaining the `umem_id`, it will be used (plus `umem_offset` inside that UMEM) to reference a physical address range in each command that points to memory. Physical addresses are used to refer to the following object types (but not only):

- WQ/CQ buffers (holding WQEs / CQEs)
- Doorbell records
- Data buffers, by Mkeys with `access_mode=MTT`

A significant usage of this feature is to allow user space drivers that operate the device directly using PRM commands (as explained in Section 8.27.1, “UCTX - User Context,” on page 539 above) to specify DMA buffers for the device without knowing their physical addresses (which is the common case with user space processes).

To achieve that, the user process should first register its memory with the kernel. In turn, the kernel will (1) extract, pin and DMA-map the physical addresses and (2) register them with the device using `CREATE_UMEM` command, and return the `umem_id` to the user. Afterwards, the user space driver can embed the `umem_id` in its PRM commands that will be passed to the device directly, without kernel intervention.

In each command that requires a PASs list (for example, `CREATE_QP`; `CREATE_CQ`; WQ context; etc.), these PASs should be replaced with a `umem_id` and a `umem_offset` that specify the start offset in the allocated buffer. It is also required to replace the DBR address (Doorbell record).

Note: When adding a reference to the buffer using `umem_id` instead of PASs, the value of all fields describing the buffer will become “N/A” (such as `page_offset` and others).

8.27.3 NVMe Device Emulation Object

TBD - Oren to provide updated text

This context is used to read/write the registers BAR of an emulated device.

In case of writing to these registers, a Device Emulation Event will be generated (See Section 8.20.12.10, “Device Emulation Event,” on page 443).

The size of the Bar registers is reported in HCA_CAP.Device_Emulation_CAPs.registers_size (See Table 270, “NVMe Device Emulation Object Format,” on page 541).

Note that read and write of this context is allowed only for device emulation manager (HCA_CAP.nvme_device_emulation_manager).

Table 270 - NVMe Device Emulation Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											0h-04h
																vhca_id																08h
resources_on_emulation_manager_enabled																																0Ch
																																10h
																																14h-1Ch
																device_pci_parameters (Table 326, “Device PCI Parameters Format,” on page 596)																20h-2Ch
																registers_data[0]																30h
																registers_data[1]																34h

Table 270 - NVMe Device Emulation Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										38h...

Table 271 - NVMe Device Emulation Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. Bit 0: register_data
08h	15:0	vhca_id	vhca identifier of the device emulation. Used only by device emulation manager (HCA_CAP.nvme_device_emulation_manager==1).
0Ch	31	enabled	The field indicates whether the NVMe emulated function is enabled and ready to be used.
	30	resources_on_emulation_manager	If set, device will allocate resources on emulation manager instead of emulated function. This means that emulation resources must not be allocated on behalf by a vhca_tunnel. Supported only when HCA_CAP.resources_on_nvme_emulation_manager==1.
10h	31:0	counter_set_id	Device Emulation counter set object's ID. 0 means that none of the counters set is used.
20h-2Ch	128	device_pci_parameters(Table 326, "Device PCI Parameters Format," on page 596)	
30h-..	31:0	registers_data[...]	Bar registers data. size reported in HCA_CAP.Device_Emulation_CAPs.registers_size

8.27.4 NVME SQ - NVME Emulation SQ Object

This object represents an emulation context for an NVMe SQ in the designated NVMe emulation function.

This object supports *create*, *destroy* and *modify* operations. ~~Modify~~ only supports updating the QP assigned for this NVMe SQ.

Parameters in this object define the location of the NVMe SQ, its doorbell, the NVMe CQ to report completions and what kind of protocol conversion should be done.

When creating an NVMe SQ object, it may not have a QP assigned to it (QPN is 0). This state is perfectly fine, and the NVMe SQ is considered operational, i.e., host may start posting requests on it. These requests, however, cannot be delivered anywhere and will be queued internally by the device, until the NVMe SQ is considered full.

Software may assign and de-assign QP to an NVMe SQ object during the lifetime of the NVMe SQ. When NVMe SQ is created, it is always created with no QP assigned (qpn=0).

Assigning a new QP is only allowed when no QP has been previously assigned (qpn was 0).

De-assigning is done by setting qpn=0. When done, the device will flush all pending IOs on the QP. QP will be left in RTS state, and will be ready to be used again, if needed. QP will return under software ownership, in the same state it was in before it was assigned to NVMe SQ.

QP assigned to NVME_SQ can be created on any rdma capable function available for the embedded CPU, if the device supports has CAP.resources_on_nvme_emulation_manager feature. Otherwise, the QP must be created using a vhca_tunnel on the emulated function itself.

Once assigned to an NVME_SQ, the QP should no longer be used by the software (until it is de-assigned or this object is destroyed).

The assigned QP should be an RC QP in the RTS state.

When destroying an NVMe SQ object with a QP assigned to it, the QP will transition to error state and will be flushed.

A QP assigned to NVMe SQ object should not be destroyed by software before de-assigning it.

Note that this context is allowed to be used only for device emulation manager (HCA_ CAP.nvme_device_emulation_manager).

Table 272 - NVME_SQ Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												0h-04h
modify_field_select																													08h				
device_emulation_id																													08h				
																													0Ch				
net-work_stat e		offload_ type		qpn																									10h				
qpn_vhca_id																log_n- vme_page_ size		msix_vector						14h									

Table 272 - NVME_SQ Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0													18h-1Ch
nvme_base_addr																																		
max_transaction_size								nvme_log_entry_size								nvme_num_of_entries																		
nvme_doorbell_offset																																		
counter_set_id																																		
emulated_device_dma_mkey																																		
nvme_cq_id																																		

Table 273 - NVME_SQ Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. Bit 0: QPN - modify QPN and qpn_vhca_id Bit 1: network_state
08h	31:0	device_emulation_id	Device emulation Object id this is belong to.

Table 273 - NVME_SQ Object Fields

Offset	Bits	Name	Description
10h	31:28	network_state	Network NVME SQ state. 0x0: INIT 0x1: RDY 0x2: ERROR
	27:24	offload_type	Offload type to be performed by device emulation. 0x0: SQE - supported only when HCA_CAP.DEVICE_EMULATION_CAP.nvme_offload_type_SQE == 1. 0x1: DOORBELL_ONLY - supported only when HCA_CAP.DEVICE_EMULATION_CAP.nvme_offload_type_doorbell_only == 1. 0x2: NVMF_CC - NVMF command Capsule.HCA_CAP.DEVICE_EMULATION_CAP.nvme_offload_type_command_capsule == 1.
	23:0	qpn	A QP to be used for delivering the NVMe SQ submissions. Setting this to 0 means no QP is assigned to the NVMe SQ.
14h	31:16	qpn_vhca_id	VHCA identifier where QP is created.
	12:8	log_nvme_page_size	Log(base 2) of the page size host is working with the NVMe device in granularity of 4KB. i.e. page size in bytes is $2^{(12+\log\_nvme\_page\_size)}$. Required for offload_type=0x2.
	7:0	msix_vector	MSI-X vector to raise when receiving RMDA-WRITE with immediate. Required only when offload_type=0x1 (doorbell only).
18h-1Ch	64	nvme_base_addr	sqe base physical address. Valid only when offload_type==0 (SQE).
20h	31:24	max_transaction_size	Device will verify that each SQE does not exceeds this size in granularity of $2^{\log\_nvme\_page\_size}$. Required for offload_type=0x2.
	23:16	nvme_log_entry_size	log(base 2) of the sqe entry size. Valid only when offload_type==0 (SQE).
	15:0	nvme_num_of_entries	sqe number of entries. Valid only when offload_type==0 (SQE).
24h	31:0	nvme_doorbell_offset	Doorbell offset from the beginning of BAR0 in granularity of bytes.
28h	31:0	counter_set_id	NVME SQ counter set object's ID. 0 means that none of the counters set is used.

Table 273 - NVME_SQ Object Fields

Offset	Bits	Name	Description
30h	31:0	emulated_device_dma_mkey	A memory key that can be used in RDMA read/write operations to access the emulated function host's physical memory. Valid only for QUERY and after QP is assigned to the NVME_SQ.
34h	31:0	nvme_cq_id	The NVMe CQ this SQ should report to. Required only for offload_type=0x2.

8.27.5 NVME_NAMESPACE OBJEC

Used only in NVMe-oF Command Capsule mode (CC).

T

Table 274 - NVME_NAMESPACE Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												0h-04h
modify_field_select																																	
device_emulation_id																														08h			
src_nsid																														0Ch			
dst_nsid																														10h			
								lba_size								metadata_size														14h			
counter_set_id																														18h			
																														1Ch-FCh			

Table 275 - NVME_NAMESPACE Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.

Table 275 - NVME_NAMESPACE Object Fields

Offset	Bits	Name	Description
08h	31:0	device_emulation_id	
0Ch	31:0	src_nsid	
10h	31:0	dst_nsid	
14h	23:16	lba_size	
	15:0	metadata_size	Must be power of 2.
18h	31:0	counter_set_id	0 indicates no counter set is connected to this namespace.

8.27.6 NVME CQ - NVME Emulation CQ Object

The NVME_CQ object represents an emulation context for a NVMe CQ in the designated NVMe emulation function. This NVMe CQ is expected to be referenced tom when creating an NVMe SQ.

This object supports *create* and *destroy* operations. Trying to destroy an NVMe CQ object while there are NVMe SQs reporting to it results in an error.

Note that this context is allowed to be used for device emulation manager only (HCA_CAP.nvme device emulation manager).

Table 276 - NVME_CQ Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																												0h-04h					
device_emulation_id																												08h					
																												0Ch					
				offload_ type																								10h					
																							msix_vector					14h					
nvme_base_addr																												18h-1Ch					
								nvme_log_entry_size						nvme_num_of_entries														20h					

Table 276 - NVME_CQ Object Format

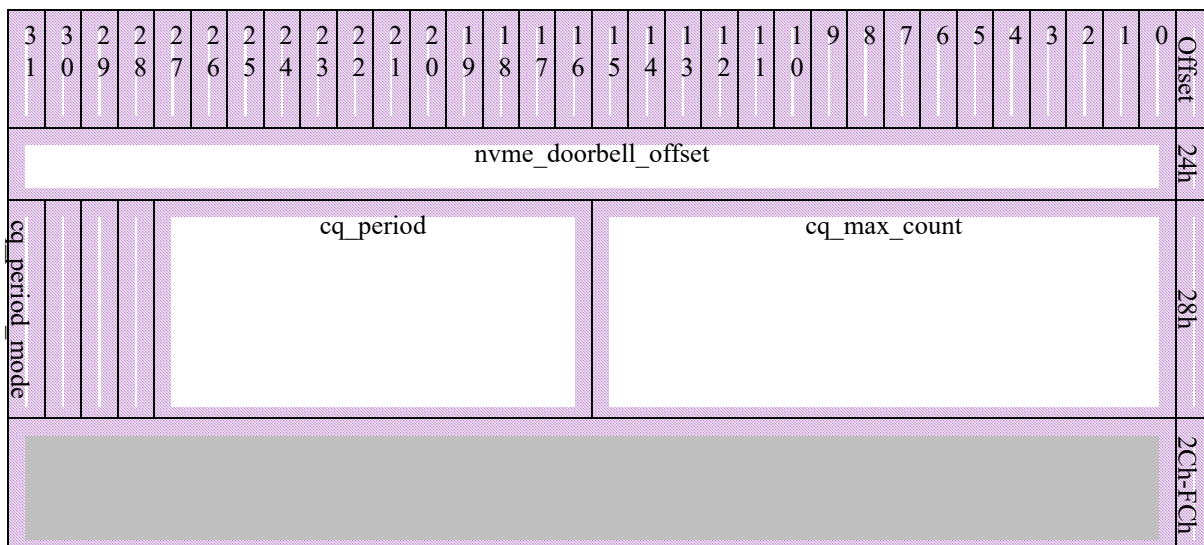


Table 277 - NVME_CQ Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
08h	31:0	device_emulation_id	Device emulation Object id this is belong to.
10h	27:24	offload_type	Offload type to be performed by device emulation. 0x0: CQE 0x2: NVMe_CQE_CC - NVMe CQE Capsule.
14h	7:0	msix_vector	MSI-X vector that should be generate when an NVMe completion is reported.
18h-1Ch	64	nvme_base_addr	cq base physical address.
20h	23:16	nvme_log_entry_size	log(base 2) of the cq entry size.
	15:0	nvme_num_of_entries	cq number of entries.
24h	31:0	nvme_doorbell_offset	Doorbell offset from the beginning of BAR0 in granularity of bytes.

Table 277 - NVME_CQ Object Fields

Offset	Bits	Name	Description
28h	31	cq_period_mode	0: upon_event - cq_period timer restarts upon event generation. 1: upon_cqe - cq_period timer restarts upon completion generation.
	27:16	cq_period	Event Generation moderation timer in 1 usec granularity, see See Section 8.20.18, “Completion Event Moderation,” on page 505. 0 - CQ moderation disabled
	15:0	cq_max_count	Event Generation Moderation counter, see See Section 8.20.18, “Completion Event Moderation,” on page 505. 0 - CQ moderation disabled

8.27.7 NVME_CTRL_COUNTERS Object

This object is used to create a counters set which is connected to a device emulation object to count ctrl operations on this device.

Multiple device emulation objects may use the same counter set.

To clear counters, a modify with value 0 on the counters besides setting the field select can be used.

Table 278 - NVME_CTRL_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											0h-04h
modify_field_select																												08h-0Ch				
																												10h				
data_read																												14h				
data_write																												18h				
cmd_read														cmd_write														1Ch				
error_cqes																												20h				
flrs																												24h				
bad_doorbells																																

Table 278 - NVME_CTRL_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											
integrity_errors																fabric_errors																28h
busy_time																power_cycle																2Ch
power_on_hours																unsafe_shutdowns																30h
error_information_log_entries																																34h
																																38h-FCh

Table 279 - NVME_CTRL_COUNTERS Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (and also indicates counter is supported - 1 means modifiable), and in MODIFY which fields to modify.
10h	31:0	data_read	Data read in granularity of 512B
14h	31:0	data_write	Data write in granularity of 512B
18h	31:16	cmd_read	Read commands
	15:0	cmd_write	Write commands
1Ch	31:0	error_cqes	Number of error CQEs reported to host
20h	31:0	flrs	Number of FLRs
24h	31:0	bad_doorbells	Doorbells to non existing queue
28h	31:16	integrity_errors	CRC errors
	15:0	fabric_errors	Errors from remote target
2Ch	31:16	busy_time	Minutes of work
	15:0	power_cycle	Number of power cycles
30h	31:16	power_on_hours	Total hours powered
	15:0	unsafe_shutdowns	Unsafe shutdowns
34h	31:16	error_informa- tion_log_entries	Error Information Log Entries

8.27.8 NVME_NAMESPACE_COUNTERS Object

This object is used to create a counters set which can be connected to an NVME_NAMESPACE object to count operations on it.

Multiple NVME_NAMESPACE objects may use the same counter set.

To clear counters, the software can use a MODIFY command with value 0 on the counters while setting the modify_field_select.

Table 280 - NVME_NAMESPACE_COUNTERS Object Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																																0h-04h
																																08h-0Ch
data_read																																10h
data_write																																14h
cmd_read																cmd_write																18h
error_cqes																																1Ch
integrity_errors																fabric_errors																20h
busy_time																power_cycle																24h
power_on_hours																unsafe_shutdowns																28h
error_information_log_entries																																2Ch
																																30h-FCh

Table 281 - NVME_NAMESPACE_COUNTERS Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (and also indicates counter is supported - 1 means modifiable), and in MODIFY which fields to modify.
10h	31:0	data_read	Data read in granularity of 512B
14h	31:0	data_write	Data write in granularity of 512B
18h	31:16	cmd_read	Read commands
	15:0	cmd_write	Write commands
1Ch	31:0	error_cqes	Number of error CQEs reported to host
20h	31:16	integrity_errors	CRC errors
	15:0	fabric_errors	Errors from remote target
24h	31:16	busy_time	Minutes of work
	15:0	power_cycle	Number of power cycles
28h	31:16	power_on_hours	Total hours powered
	15:0	unsafe_shutdowns	Unsafe shutdowns
2Ch	31:16	error_information_log_entries	Error Information Log Entries

8.27.9 NVME_SQ_COUNTERS Object

This object is used to create a counters set which can be connected to an NVME_SQ object to count operations on it.

Multiple NVME_SQ objects may use the same counter set.

To clear counters, the software can use a MODIFY command with value 0 on the counters while setting the modify_field_select.

Table 282 - NVME_SQ_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										0h-04h
modify_field_select																															

Table 282 - NVME_SQ_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										08h-0Ch
data_read																											10h				
data_write																											14h				
cmd_read													cmd_write														18h				
error_cqes																											1Ch				
integrity_errors													fabric_errors														20h				
busy_time													power_cycle														24h				
power_on_hours													unsafe_shutdowns														28h				
error_information_log_entries																											2Ch				
nvme_pci_errors																											30h				
																											34h-FCh				

Table 283 - NVME_SQ_COUNTERS Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (and also indicates counter is supported - 1 means modifiable), and in MODIFY which fields to modify.
10h	31:0	data_read	Data read in granularity of 512B
14h	31:0	data_write	Data write in granularity of 512B
18h	31:16	cmd_read	Read commands
	15:0	cmd_write	Write commands
1Ch	31:0	error_cqes	Number of error CQEs reported to host

Table 283 - NVME_SQ_COUNTERS Fields

Offset	Bits	Name	Description
20h	31:16	integrity_errors	CRC errors
	15:0	fabric_errors	Errors from remote target
24h	31:16	busy_time	Minutes of work
	15:0	power_cycle	Number of power cycles
28h	31:16	power_on_hours	Total hours powered
	15:0	unsafe_shutdowns	Unsafe shutdowns
2Ch	31:16	error_informa- tion_log_entries	Error Information Log Entries
30h	31:0	nvme_pci_errors	PCI errors accessing p2p

8.27.10SW ICM Block Object

The object used to allocate a contiguous ICM range, accessible to SW. [See TBD](#)

Table 284 - SW ICM Block Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												0h-04h
modify_field_select																														08h			
																												log_sw_icm_size		08h			
																												0Ch					
sw_icm_start_address																												10h-14h					
																												18h-3Ch					

Table 285 - SW ICM Block Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
08h	7:0	log_sw_icm_size	Log (base 2) of the allocated range size. Must be aligned with log_sw_icm_alloc_granularity in QUERY_HCA_CAP.
10h-14h	64	sw_icm_start_address	ICM start address of the allocated range. Must be aligned with log_sw_icm_alloc_granularity in QUERY_HCA_CAP.

8.27.11 Flow Meter Object

Table 286 - Flow Meter Object Format

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																														0h-04h		
active				return_reg_id				table_type																							08h	
									destination_table_id																				0Ch			
																																10h
																																14h-1Ch
flow_meter_parameters (See Table 288, “Flow Meter Parameters Format,” on page 558)																																20h-3Ch
																																40h-6Ch
sw_steering_icm_address_rx																																70h-74h
sw_steering_icm_address_tx																																78h-7Ch
																																80h-FCh

Table 287 - Flow Meter Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. Bit 0: Active Bit 1: CBS - affects cbs_exponent and cbs_mantissa Bit 2: CIR - affects cir_exponent and cir_mantissa Bit 3: EBS - affects ebs_exponent and ebs_mantissa Bit 4: EIR - affects eir_exponent and eir_mantissa
08h	31	active	When set, the Monitor is considered connected to at least one Flow and should be monitored
	27:24	return_reg_id	Index of Register C, where the packet Color will be set after passing through the Meter. valid values are according to QUERY_H-CA_CAP.flow_meter_reg_id. the result will be set in the 8 LSB of the register
	23:16	table_type	Table's role in packet processing for the tables pointing to/from the Meter 0x0: NIC_RX 0x1: NIC_TX 0x2: ESwitch_egress_ACL 0x3: ESwitch_ingress_ACL 0x4: ESwitch_FDB 0x5: Nic_Sniffer_Rx 0x6: Nic_Sniffer_Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
0Ch	23:0	destination_table_id	Destination Table to which packet would be redirected after passing through the Meter.
20h-3Ch	256	flow_meter_parameters (See Table 288, "Flow Meter Parameters Format," on page 558)	Flow Meter parameters. See Table 288, "Flow Meter Parameters Format," on page 558
70h-74h	64	sw_steering_icm_address_rx	ICM address used by SW managed steering to point traffic to this Meter. Valid For Rx Table Types only, and for FDB table type Rx domain. Valid for Query operation only.
78h-7Ch	64	sw_steering_icm_address_tx	ICM address used by SW managed steering to point traffic to this Meter. Valid For Tx Table Types only, and for FDB table type Tx domain. Valid for Query operation only.

Table 288 - Flow Meter Parameters Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h	
bucket_overflow valid		Start_color		both_buckets_on_green		meter_mode																										
		cbs_exponent		cbs_mantissa											cir_exponent				cir_mantissa				08h									
																																0Ch
		ebs_exponent		ebs_mantissa										eir_exponent				eir_mantissa				10h										
																																14h-1Ch

Table 289 - Flow Meter Parameters Fields

Offset	Bits	Name	Description
00h	31	valid	Valid Flow Meter Context. Must be set to 0x1
	30	bucket_overflow	Tokens overflow from C bucket to E bucket on return(relevant for MEF and srTCM) Must be set to 0x1
	29:28	Start_color	Initial Color applied to packets passing through the Meter. 0x0: RED, 0x1: YELLOW, 0x2: GREEN, 0x3: UNDEFINED. Must be set to GREEN
	27	both_buckets_on_green	Check tokens in both buckets if precolor on green packet – RFC 4115. Must be set to 0x0
	26:25	meter_mode	Indicates the measurement performed by the meter. 0x0: BYTES_IP_LENGTH - packet bytes as indicated by IP.length 0x1: BYTES_CALC_WITH_L2 - packet bytes as calculated by the device including L2 headers 0x2: BYTES_CALC_WITH_L2_IPG - packet bytes as measured by the device including L2 headers and IPG 0x3: NUM_PACKETS - number of packets must be set to 0x0 or 0x1
08h	28:24	cbs_exponent	Exponent value of Committed Bucket Size
	23:16	cbs_mantissa	Mantissa value of Committed Bucket Size. Bucket size is given in token units and is calculated by $cbs\_mantissa * 2^{cbs\_exponent}$
	12:8	cir_exponent	Exponent value of Committed Information rate.
	7:0	cir_mantissa	Mantissa of Committed Information rate. Rate is given in bit per second and is calculated by $8 * (10^9) * cir\_mantissa * 2^{(-1 * cir\_exponent)}$
10h	28:24	ebs_exponent	Exponent value of Excess Bucket Size
	23:16	ebs_mantissa	Mantissa value of Excess Bucket Size. Bucket size is given in token units and is calculated by $ebs\_mantissa * 2^{ebs\_exponent}$. Must be set to 0x0
	12:8	eir_exponent	Exponent value of Excess Information rate Must be set to 0x0
	7:0	eir_mantissa	Mantissa of Excess Information rate. Rate is given in bit per second and is calculated by $8 * (10^9) * eir\_mantissa * 2^{(-1 * eir\_exponent)}$

8.27.12 Flow Sampler Object

The purpose of the FLOW_SAMPLER object is to 'sample' all packets steered to it, at a given 'ratio' (statistical approximation).

All the packets continue to the default_table_id destination, beside any replication of the sampled packets to the sample_table_id destination.

The replicated sampled packets must be destined to a 'terminating flow table' destination which can add any action and forward to a TIR or VPORT.

Packets in each destination are delivered in order, while ordering between the default and sampled destinations is not guaranteed.

This object respects the flow table level guidelines.

FDB table_type limitations:

- The termination table on the sample destination path must have the vport of the eswitch_manager and no additional actions
- The object will pass RegC0 from input to both sampler and default output destination

Table 290 - Flow Sampler Object Format

<u>3</u> <u>1</u>	<u>3</u> <u>0</u>	<u>2</u> <u>9</u>	<u>2</u> <u>8</u>	<u>2</u> <u>7</u>	<u>2</u> <u>6</u>	<u>2</u> <u>5</u>	<u>2</u> <u>4</u>	<u>2</u> <u>3</u>	<u>2</u> <u>2</u>	<u>2</u> <u>1</u>	<u>2</u> <u>0</u>	<u>1</u> <u>9</u>	<u>1</u> <u>8</u>	<u>1</u> <u>7</u>	<u>1</u> <u>6</u>	<u>1</u> <u>5</u>	<u>1</u> <u>4</u>	<u>1</u> <u>3</u>	<u>1</u> <u>2</u>	<u>1</u> <u>1</u>	<u>1</u> <u>0</u>	<u>9</u>	<u>8</u>	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>	Offset								
modify_field_select																												0h-04h												
<u>table_type</u>								<u>level</u>								ignore_flow_level												08h												
sample_ratio																												0Ch												
								<u>sample_table_id</u>																				10h												
								<u>default_table_id</u>																				14h												
								<u>sw_steering_icm_address_rx</u>																												18h-1Ch				

Table 290 - Flow Sampler Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											20h-24h
																												28h-3Ch				

Table 291 - Flow Sampler Object Fields

Offset	Bits	Name	Description
00h-04h	64	<u>modify_field_select</u>	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
08h	31:24	<u>table_type</u>	see CREATE_FLOW_TABLE.table_type
	23:16	<u>level</u>	see CREATE_FLOW_TABLE.flow_table_context.level
	0	<u>ignore_flow_level</u>	When set, the default_table_id may point to Flow Table with any Level
0Ch	31:0	<u>sample_ratio</u>	the probability for a packet to be sampled by the sampler is 1/ <u>sample_ratio</u>
10h	23:0	<u>sample_table_id</u>	Sampled packets destination. Must be a termination table_id from same table_type
14h	23:0	<u>default_table_id</u>	All packets destination table_id from same table_type
18h-1Ch	64	<u>sw_steering_icm_address_rx</u>	ICM address used by SW managed steering to point traffic to this Meter. Valid For Rx Table Types only, and for FDB table type Rx domain. Valid for Query operation only.
20h-24h	64	<u>sw_steering_icm_address_tx</u>	ICM address used by SW managed steering to point traffic to this Meter. Valid For Tx Table Types only, and for FDB table type Tx domain. Valid for Query operation only.

8.27.13 GENEVE TLV OPTION Object

Geneve protocol supports optional headers in a TLV format. The device supports parsing the optional headers, and using them for packet steering process. In order to use a specific optional header for packet steering, a Geneve TLV Option Object must be created, specifying the option's class, type and length. Once created, this option is associated with an index (`geneve_option_fte_index`), indicating which steering fields should be used to reference it (`geneve_tlv_option_<index>_data`, `geneve_tlv_option_<index>_exist`).

The amount of Geneve options that can be matched is limited by `max_geneve_tlv_options`, and the maximal length of a matched option data is limited by `max_geneve_tlv_option_data_len`. Geneve option with type/class/length=0x0 is not supported.

When matching the existence and data value of a Geneve option, the following limitations apply:

- If data value is matched (`mask!=0`), the existence of the option must be matched as well, and the data value must be non-zero.
- Matching only for the existence of the option is only supported for options with data length 0x0
- Matching on the option data is not supported if data length is 0x0

Table 292 - GENEVE TLV OPTION Object Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																																0h-04h
																								geneve_option_fte_index				08h				
option_class																option_type										option_data_length		0Ch				
																																10h-3Ch

Table 293 - GENEVE TLV OPTION Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.

Table 293 - GENEVE TLV OPTION Fields

Offset	Bits	Name	Description
08h	7:0	geneve_option_fte_index	Read Only field describing the option ID that should be used when matching the option data values by FTE.
0Ch	31:16	option_class	
	15:8	option_type	
	4:0	option_data_length	Data length in granularity of four bytes. Must be <= HCA_CAP.max_geneve_tlv_option_data_len.

8.27.14DEK Object

Table 294 - DEK Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h-04h
modify_field_select																												00h-04h			
state																				key_size		has_keytag			key_purpose		08h				
								pd																				0Ch			
																												10h-2Ch			
opaque																												30h-34h			
																												38h-3Ch			
key																												40h-BCh			
																												C0h-FCh			

Table 295 - DEK Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.

Table 295 - DEK Fields

Offset	Bits	Name	Description
08h	31:24	state	Key state, Valid for Query command only. 0x0: READY - the key state when first created 0x1: ERROR - key unusable, need to destroy and re-create this key
	11:8	key_size	Key size 0x0: SIZE_128 0x1: SIZE_256
	7	has_keytag	If set, then key should include a 8B keytag that is compared to MKEY.BSF.keytag at runtime.
	3:0	key_purpose	0x1: TLS 0x2: IPSEC 0x3: AES_XTS 0x4: MACSEC - (internal) 0x5: GCM_mem2mem - (internal)
0Ch	23:0	pd	Protection domain. valid only when key_type==DEK.
30h-34h	64	opaque	Plaintext metadata to describe the key.
40h-BCh	1024	key	For key size of 128b the MSBs are reserved. Actual size and layout of this field depends on the provided key_purpose, key_size and has_keytag fields, as well as the wrapped_import_method capability for the respective engine. See DEK key field construction summary in Table 296, “DEK Key Field Construction Summary,” on page 565 below.

Table 296 - DEK Key Field Construction Summary

Key Purpose	Key Size	Has Keytag	Import Method	Key layout
TLS	128	No	Plaintext	pad_128b + key_128b
TLS	256	No	Plaintext	key_256b
TLS	128	No	Wrapped	ENC(iv_64b + key_128b)
TLS	256	No	Wrapped	ENC(iv_64b + key_256b)
AES-XTS	128	Yes	Plaintext	key1_128b + key2_128b +keytag_64b

Table 296 - DEK Key Field Construction Summary

Key Purpose	Key Size	Has Keytag	Import Method	Key layout
AES-XTS	256	Yes	Plaintext	key1_256b + key2_256b +keytag_64b
AES-XTS	128	Yes	Wrapped	ENC(iv_64b + key1_128b +key2_128b + keytag_64b)
AES-XTS	256	Yes	Wrapped	ENC(iv_64b + key1_256b +key2_256b + keytag_64b)
IPSEC	128	No	Plaintext	pad_128b + key_128b
IPSEC	256	No	Plaintext	key_256b
IPSEC	128	No	Wrapped	ENC(iv_64b + key_128b)
IPSEC	256	No	Wrapped	ENC(iv_64b + key_256b)

Notes:

Note: pad_x is padding of zero bits

Note: Layout begins at the beginning of the 1024 field

Note: The rest of the 1024 field should be reserved with zero bits

Note: When import method is “Wrapped”, its AES would be “keywrap”

8.27.15IMPORT_KEY Object

Table 297 - IMPORT_KEY Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	modify_field_select											00h-04h
state																			key_size								08h						
																																0Ch-3Ch	
key																																40h-BCh	
																																C0h-FCh	

Table 298 - IMPORT_KEY Object Fields

Offset	Bits	Name	Description	Access (for registers only)
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.	INDEX
08h	31:24	state	Key state, Valid for Query command only. 0x0: READY - the key state when first created 0x1: ERROR - key unusable, need to destroy and re-create this key	INDEX
	11:8	key_size	Key size 0x0: SIZE_128 0x1: SIZE_256	INDEX

Table 298 - IMPORT_KEK Object Fields

Offset	Bits	Name	Description	Access (for registers only)
40h- BCh	1024	key	For key size of 128b the MSBs are reserved. Actual size and layout of this field described in Table 299, “IMPORT_KEK Field Construction Summary,” on page 568 below.	<u>INDEX</u>

Table 299 - IMPORT_KEK Field Construction Summary

Key Size	Import Method	Key layout
128	Wrapped	ENC(iv_64b + key_128b)
256	Wrapped	ENC(iv_64b + key_256b)

Notes:

- Note:** pad_x is padding of zero bits
- Note:** Layout begins at the beginning of the 1024 field
- Note:** The rest of the 1024 field should be reserved with zero bits
- Note:** When import method is “Wrapped”, its AES would be “keywrap”

8.27.16 CREDENTIAL Object

Required for wrapped_import_method. Credentials are used for any crypto operation in this mode, regardless of the key usage. At least one officer credential is required.

Table 300 - CREDENTIAL Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	modify_field_select										00h-04h
state																		credential_role								08h						
																												0Ch-3Ch				
credential																												40h-6Ch				
																												70h-FCh				

Table 301 - CREDENTIAL Fields

Offset	Bits	Name	Description	Access (for registers only)
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.	<u>INDEX</u>
08h	31:24	state	Credential state, Valid for Query command only. 0x0: READY - the Credential state when first created 0x1: ERROR - Credential unusable, need to destroy and re-create this credential.	<u>INDEX</u>
	7:0	credential_role	0x0: role_officer - Officer role. 0x1: role_user - User role.	<u>INDEX</u>
40h-6Ch	384	credential	The actual secret. Should be wrapped by import_KEK since credentials are only used in wrapped import method. Size includes 8 bytes IV for wrapping.	<u>INDEX</u>

8.27.17 CRYPTO_LOGIN Object

Login to the device as crypto user or crypto officer.

Required in order to perform any crypto related control operations when the respective engine wrapped_import_method is set. Also defines the import_KEK that will be used for further crypto commands.

Destroying the object means logout.

Table 302 - CRYPTO_LOGIN Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-04h
modify_field_select																												08h-0Ch				
								credential_pointer																				10h				
								session_import_kek_ptr																				14h				
																												18h-3Ch				
credential																												40h-6Ch				
																												70h-FCh				

Table 303 - CRYPTO_LOGIN Fields

Offset	Bits	Name	Description	Access (for registers only)
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.	INDEX
10h	23:0	credential_pointer	Credential pointer	INDEX
14h	23:0	session_import_kek_ptr	Import_KEK pointer to be used for all secured information communicated in crypto commands (key fields), including the provided credential in this command	INDEX
40h-6Ch	384	credential	The credential secret, wrapped by the import_KEK indicated in this command. Size includes 8 bytes IV for wrapping	INDEX

8.27.18 VIRTIO_NET_Q Object

This object is supported only when `HCA_CAP.general_obj_types.VIRTQ_NET_Q==1`.

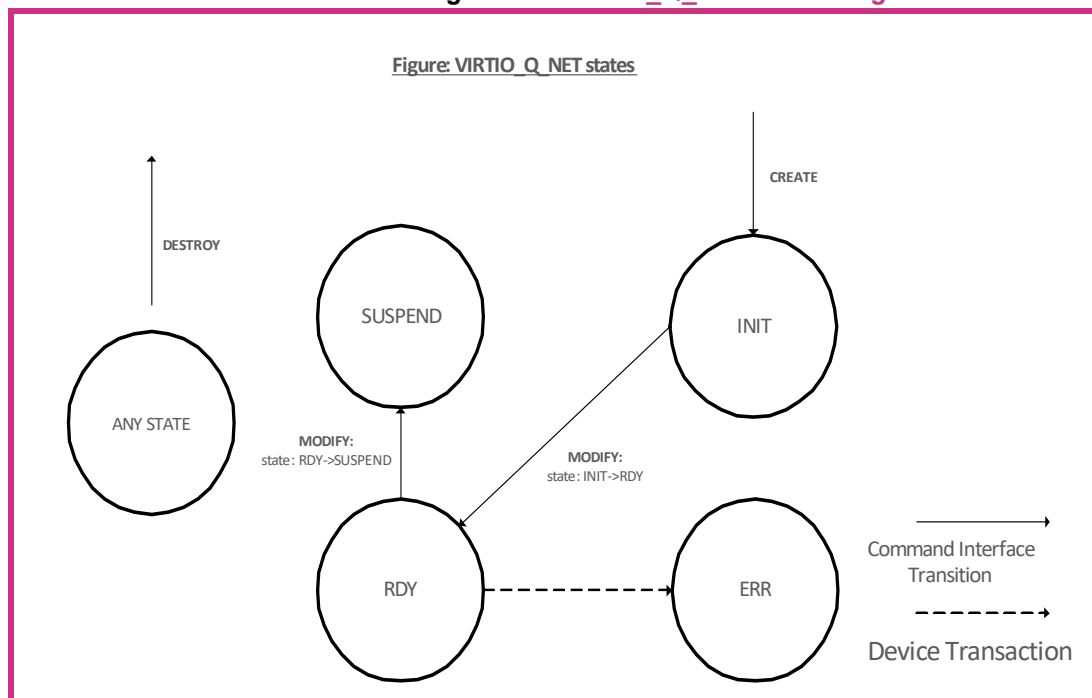
The VIRTIO_NET_Q object represents an emulated context for a VIRTIO_NET Virtqueue which was created and managed by a VIRTIO_NET driver as defined in VIRTIO Specification.

Software should provide all information of the VIRTIO_NET Virtqueue it wants to accelerate when creating the context, and by modifying the state as described in the conditions below.

When the VIRTIO_NET_Q is in operational state (RDY) the device is assumed to be the only entity which performs the VIRTIO_NET Device-side data path engine, and communicates directly with the VIRTIO_NET driver via VIRTIO doorbell kicks, MSI-X interrupts, updating and reading the used/available/descriptor rings, etc..as described in the VIRTIO specification.

VIRTIO_NET_Qs can be attached to a TIR/RQT to be used for steering purposes, meaning that a VIRTIO_NET_Q object ID can be used instead of RQN, when creating a TIR (inline_rqn) or a RQT.

Figure 44: VIRTIO_Q_NET State Diagram



When creating a VIRTIO_NET_Q object, SW should provide the required UMEM buffers needed internally to support the emulation. The number of umem buffers and their size are reported in VIRTIO_NET_EMULATION_Capabilities and VDPA_EMULATION_Capabilities.

After successful creation the VIRTQ object is in INIT state. SW then should modify the object and move it to RDY state. SW may also modify the object's state to SUSPEND, which will result in stopping all I/O operations to be executed by the device. SW can then query the object for device information (mainly hw_available_idx and hw_used_idx) and it's guaranteed to have all information valid without any I/O operation pending by the device.

If modify object success (using MODIFY_GENERAL_OBJ), SW can assume that it also moved to the required state.

In case device encounter an error in a VIRTIO_NET_Q operation, it moves the state to ERR and issues a corresponding Object Change Event. Then SW can query the object to get the detailed error as reported by syndrome.

For further information on how to use this object, please refer to Section 20.7, “VIRTIO Emulation Off-load,” on page 1008.

Table 304 - VIRTIO_NET_Q Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-04h	
modify_field_select																															00h-04h		
																															08h		
tirn_or_qpn_vhca_id																																0Ch	
tso_ipv4	tso_ipv6	tx_csum	rx_csum									dirty_bitmap_dump_enable	vhost_log_page																state				10h
error_type								tirn_or_qpn																							14h		
dirty_bitmap_mkey																															18h		
dirty_bitmap_size																															1Ch		
dirty_bitmap_addr																															20h-24h		
hw_available_index															hw_used_index																	28h	
																															2Ch-3Ch		
virtio_q_context(Table 306, “VIRTIO_Q Format,” on page 575)																															40h-BCh		

Table 305 - VIRTIO_NET_Q Fields Description

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1means modifiable), and in MODIFY which fields to modify. Bit 0 : state Bit 3 : dirty_bitmap_params - For dirty_bitmap_mkey,dirty_bitmap_addr, dirty_bitmap_size and vhost_log_page Bit 4 : dirty_bitmap_dump_enable
0Ch	31:16	~~tisn_or_qpn_vhca_id~~	~~VHCA identifier where QP/TISN provided in tisn_or_qpn field is created.~~ ~~Reserved when tisn_or_qpn not provided.vhca_id the virtio queue is affiliated with. valid only when virtio_q.full_emulation is set, otherwise reserved.~~
10h	31	tso_ipv4	When set queue will support virtio TSO for IPV4 packets. Supported only when HCA_CAP.virtio_cap.tso_ipv4.
	30	tso_ipv6	When set queue will support virtio TSO for IPV6 packets. Supported only when HCA_CAP.virtio_cap.tso_ipv6.
	29	tx_csum	When set queue will supports virtio csum offload for Tx packets. Supported only when HCA_CAP.virtio_cap.tx_csum. Must be set in case tso_ipv* is set
	28	rx_csum	When set queue will supports virtio csum offload for Rx packets. Supported only when HCA_CAP.virtio_cap.rx_csum.
	21	dirty_bitmap_dump_enable	If set, the device will log the I/O transactions of a VIRTIO_NET_Q to vHOST log dirty bitmap file. (used for live migration purposes). When set, dirty_bitmap_mkey, dirty_bitmap_size and dirty_bitmap_addr are valid.
	20:16	vhost_log_page	Log (base 2) of page size of the vHOST dirty bitmap file in granularity of 4KB.
	3:0	state	VIRTIO queue state 0x0: INIT 0x1: RDY 0x2: SUSPEND 0x3: ERR
14h	23:0	tisn_or_qpn	TIS numberor QP number to be pointed by this virtio Queue. When offload_type==eth_frame and the VIRTIO queue is Tx (according to queue index) indicates TISN. When offload_type==desc_tunneling, indicates QPN. otherwise reserved.

Table 305 - VIRTIO_NET_Q Fields Description

Offset	Bits	Name	Description
18h	31:0	dirty_bitmap_mkey	This mkey is used by the device to access the VHOST live migration log dirty bitmap file. Start address and size are indicated by dirty_bitmap_addr and dirty_bitmap_size Valid when dirty_bitmap_dump_enable==1.
1Ch	31:0	dirty_bitmap_size	Valid when dirty_bitmap_dump_enable==1.
20-24h	64	dirty_bitmap_addr	Valid when dirty_bitmap_dump_enable==1.
28h	31:16	hw_available_index	Initial value of the device emulation software available_idx to use for this VIRTIO_NET_Q.
	15:0	hw_used_index	initial value of the device emulation software available_idx to use for this VIRTIO_NET_Q.
40h-BCh	1024	virtio_q_context(Table 306, "VIRTIO_Q Format," on page 575)	Virtio Queue Context. See Table 306, "VIRTIO_Q Format," on page 575

Table 306 - VIRTIO_Q Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
virtio_q_type												event_mode		queue_index																00h			
full_emulation	virtio_version_1_0			offload_type	event_qpn_or_msix																										04h		
					doorbell_stride_index												queue_size														08h		
					device_emulation_id																												0Ch
					desc_addr																												10h-14h

Table 306 - VIRTIO_Q Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	used_addr																18h-1Ch
																					available_addr																20h-24h	
																					virtio_q_mkey																28h	
max_tunnel_desc																			error_type							2Ch												
																					umem_1_id																30h	
																					umem_1_size																38h	
																					umem_1_offset																38h-3Ch	
																					umem_2_id																40h	
																					umem_2_size																34h	
																					umem_2_offset																48h-4Ch	
																					umem_3_id																50h	
																					umem_3_size																354h	
																					umem_3_offset																58h-5Ch	
																					counter_set_id																60h	
							pd														64h																	
																																					68h-7Ch	

Table 307 - VIRTIO_Q Fields Description

Offset	Bits	Name	Description
00h	31:24	virtio_q_type	Virtio Queue type. 0x0: SPLIT 0x1: PACKED
	18:16	event_mode	Indicates event notification mode. 0x0: NO_MSIX_MODE - virtio packet will not trigger MSIX. Supported only when HCA_CAP.event_mode == NO_MSIX_MODE. 0x1: QP_MODE - virtio packet will trigger CQE on the CQ connected to event_qpn_or_msix. Supported only when HCA_CAP.event_mode == QP_MODE. 0x2: MSIX_MODE - virtio packet will trigger MSIX vector given on event_cqn_or_msix field. Supported only when HCA_CAP.event_mode == MSIX_MODE.
	15:0	queue_index	The queue_index of the queue, according to VIRTIO spec. The VIRTIO specific device virtqueue will be derived from queue_index according to the spec.

Table 307 - VIRTIO_Q Fields Description

Offset	Bits	Name	Description
04h	31	full_emulation	When set virtqueue is created for fully emulated device. emulation_object_id is valid on this case and virtio_queue_mkey should be reserved on creation.
	30	virtio_version_1_0	When set, queue will be virtio 1.0 compatible. Supported only when HCA_CAP.Virtio_CAP.virtio_version_1_0
	27:24	offload_type	Offload type to be performed by device emulation. 0x0: eth_frame – when set, device will convert virtio-net descriptors to ethernet frames. Supported only when VIRTIO- ~~NET~~ _EMULATION_CAPABILITIES.offload_type_eth_frame == 1. 0x1: descriptor_tunneling – when set, device will tunnel virtio descriptors to send operation on the provided QP. supported only when HCA_CAP.VIRTIO- ~~NET~~ _EMULATION_CAPABILITIES.offload_type_desc == 1.
	23:0	event_eqqpnr_or_msix	When event_mode QP_MODE - QP number of a QP, created by the SW, and provided to the device. After provided, the QP is in the exclusive ownership of the device. SW is not allowed to use it till the VIRTQ object destroyed. the QP should: - be a RC QP - have no SQ,RQ,UAR,DB record - have no CQs - be in RTS state and connect with a pair SW QP that is in SW ownership. The size of the SW QP should match in the number of descriptors to the size of the virtq. Device will report completions on the CQ connected to the SW QP when the VIRTIO_Q used rings is being updated. The user can, if needed, connect it to an EQ which will be used to generate the events and interrupts for the VIRTIO- ~~NET~~ _Q owner. The user should update the SW QP doorbell record after every completion consume from the CQ connected to it. When event_mode == MSIX_MODE, then indicates the MSIX vector number to trigger upon completion of virtio packet. When event_mode == NO_MSIX, then reserved.
08h	31:16	doorbell_stride_index	Doorbell stride index.
	15:0	queue_size	Number of descriptors in virtio queue.
0Ch	31:0	device_emulation_id	Device emulation id this object is associate with. Valid only when full_emulation==1
10h-14h	64	desc_addr	Descriptor table address. Used beside virtio_q_mkey. valid only when virtio_q_type == SPLIT

Table 307 - VIRTIO_Q Fields Description

Offset	Bits	Name	Description
18h-1Ch	64	used_addr	Used ring address. Used beside virtio_q_mkey valid only when virtio_q_type == SPLIT
20h-24h	64	available_addr	Available ring address. Used beside virtio_q_mkey valid only when virtio_q_type == SPLIT
28h	31:0	virtio_q_mkey	~~This is the main mkey used by the device's emulated software to access the virtqueue and data buffers pointed by it. It is used to access all the memory address pointed by the VIRTIO_Q.~~ When full_emulation is set this field is reserved on creation. Upon query, device will return the relevant mkey to access host memory. Otherwise SW should provide a mkey to be used by the device to access the virtqueue and data buffers pointed by it. It is used to access all the memory address pointed by the VIRTIO_Q.
2Ch	31:16	max_tunnel_desc	The maximum number of descriptors the SW is willing to receive on a single SEND operation. Valid only when virtq.offload_type = desc_tunneling. Must be <= HCA_CAP.max_tunnel_desc.
	7:0	error_type	Error type. 0x0: NO_ERROR 0x1: NETWORK_ERROR- One of the network component got into error state 0x2: BAD_DESCRIPTOR - Driver posted a corrupted descriptor 0x3: INVALID_BUFFER - Driver descriptor pointed to buffer that is not registered to the device 0x4: DESCRIPTOR_LIST_EXCEED_LIMIT - Driver posted descriptor list that exceeds the device limit 0x5: INTERNAL_ERROR - Device encountered internal error
30h	31:0	umem_1_id	Umem 1 identifier for internal use to perform the emulation offload.
34h	31:0	umem_1_size	Umem1 buffer size. The minimum required size should be calculated according to: buffer size = HCA_CAP.umem_1_buffer_param_a * virtio_q size + HCA_CAP.umem_1_buffer_param_b
38h-3Ch	64	umem_1_offset	Umem offset in umem for internal use to perform the emulation offload.
40h	31:0	umem_2_id	Umem 2 identifier for internal use to perform the emulation offload.

Table 307 - VIRTIO_Q Fields Description

Offset	Bits	Name	Description
44h	31:0	umem_2_size	Umem2 buffer size. The minimum required size should be calculated according to: buffer size = HCA_CAP.umem_2_buffer_param_a * virtio_q size + HCA_CAP.umem_2_buffer_param_b
48h-4Ch	64	umem_2_offset	Umem offset in umem for internal use to perform the emulation offload.
50h	31:0	umem_3_id	Umem 1 identifier for internal use to perform the emulation offload.
54h	31:0	umem_3_size	Umem3 buffer size. The minimum required size should be calculated according to: buffer size = HCA_CAP.umem_3_buffer_param_a * virtio_q size + HCA_CAP.umem_3_buffer_param_b
58h-5Ch	64	umem_3_offset	Umem offset in umem for internal use to perform the emulation offload.
60h	31:0	counter_set_id	Virtio Q Counter set ID. See
64h	23:0	pd	The Protection Domain the virtio queue is associated with. All the virtio queue DMA transactions will be validated using this pd. in case virtio_q_mkey is provided, the mkey pd should match this field [Internal] this field addition will need drivers to adopt. as agreed w/ maintainers it is OK since the upstream drivers are not yet GA.

Table 308 - VIRTIO_Q_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-04h
modify_field_select																																
																														68h-0Ch		

Table 308 - VIRTIO_Q_COUNTERS Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										10h-14h
received_desc																															18h-1Ch
completed_desc																															20h
error_cqes																															24h
bad_desc_errors																															28h
exceed_max_chain																															2Ch
invalid_buffer																															30h-7Ch

Table 309 - VIRTIO_Q_COUNTERS Object Fields Description

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1means modifiable), and in MODIFY which fields to modify. Bit 0 : received_desc Bit 1: completed_desc Bit 2: error_cqes Bit 3: bad_desc_errors Bit 4: exceed_max_chain Bit 5: invalid_buffer
10h-14h	64	received_desc	Number of available descriptors received by device
18h-1Ch	64	completed_desc	Number of available descriptors completed by the device
20h	31:0	error_cqes	Number of error CQEs received on the queue

Table 309 - VIRTIO_Q_COUNTERS Object Fields Description

Offset	Bits	Name	Description
24h	31:0	bad_desc_errors	Number of bad descriptors received
28h	31:0	exceed_max_chain	Number of chained descriptors received that exceed the max allowed chain by device
2Ch	31:0	invalid_buffer	Number of times device tried to read or write buffer that is not registered to the device

8.27.19 VIRTIO NET DEVICE EMULATION Object

Table 310 - VIRTIO NET DEVICE EMULATION Object Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										0h-04h
modify_field_select																					vhca_id							resources_on_emulation_manager		enabled	08h
																															0Ch
virtio_device																					(Table 312, "VIRTIO DEVICE Layout," on page 584)										10h-3Ch
virtio_net_config																					(Table 316, "VIRTIO NET CONFIG Layout," on page 586)										40h-4Ch

Table 310 - VIRTIO NET DEVICE EMULATION Object Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										50h-6Ch
virtio_q_configuration[0] (Table 314, “VIRTIO Q Configuration Layout,” on page 585)																															70h-8Ch
virtio_q_configuration[1] (Table 314, “VIRTIO Q Configuration Layout,” on page 585)																															90h-...
...																															

Table 311 - VIRTIO NET DEVICE EMULATION Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. 0x0: device_status 0x1: status - In virtio Config
08h	31	resources_on_emulation_manager	If set, device will allocate resources on emulation manager instead of emulated function. This means that emulation resources must not be allocated on behalf by a vhca_tunnel. Supported only when HCA_CAP.resources_on_virtio_net_emulation_manager==1.
	30	enabled	The field indicates whether the emulated function is enabled and ready to be used. Read Only.
	15:0	vhca_id	vhca identifier of the device emulation. Used only by device emulation manager (HCA_CAP.virtio_blk_device_emulation_manager==1).
10h-3Ch	384	virtio_device (Table 312, “VIRTIO DEVICE Layout,” on page 584)	See Table 312, “VIRTIO DEVICE Layout,” on page 584
40h-4Ch	128	virtio_net_config (Table 316, “VIRTIO NET CONFIG Layout,” on page 586)	See Table 316, “VIRTIO NET CONFIG Layout,” on page 586

Table 311 - VIRTIO NET DEVICE EMULATION Object Fields

Offset	Bits	Name	Description
50h- ...	256	virtio_q_configuration[...] (Table 314, “VIRTIO Q Configuration Layout,” on page 585)	List of Virtio Q Configuration (See Table 314, “VIRTIO Q Configuration Layout,” on page 585). The array size is $1 + 2 \times \text{virtio\_net\_device\_layout.max\_virtqueue\_pairs}$, where Queue index i corresponds to the <code>virtio_queue_configuration[i]</code> .

Table 312 - VIRTIO DEVICE Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-04h
device_feature																																
driver_feature																															08h-0Ch	
msix_config															num_queues																10h	
device_status								config_generation																							14h	
notify_off_multiplier																															18h	
																															1Ch	
device_pci_parameters (Table 326, “Device PCI Parameters Format,” on page 596)																															20h-2Ch	

Description as defined in virtio spec

Table 313 - VIRTIO DEVICE Fields

Offset	Bits	Name	Description
00h-04h	64	device_feature	
08h-0Ch	64	driver_feature	

Table 313 - VIRTIO DEVICE Fields

Offset	Bits	Name	Description
10h	31:16	msix_config	
	15:0	num_queues	
14h	31:24	device_status	
	23:16	config_generation	
18h	31:0	notify_off_multiplier	
20h-2Ch	128	device_pci_parameters(Table 326, “Device PCI Parameters Format,” on page 596)	

Table 314 - VIRTIO Q Configuration Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
queue_size																queue_msix_vector																
queue_enable																queue_notify_off																04h
queue_desc																																08h-0Ch
queue_driver																																10h-14h
queue_device																																18h-1Ch

Description as defined in virtio spec

Table 315 - VIRTIO Q Configuration Fields

Offset	Bits	Name	Description
00h	31:16	queue_size	
	15:0	queue_msix_vector	
04h	31:16	queue_enable	
	15:0	queue_notify_off	
08h-0Ch	64	queue_desc	
10h-14h	64	queue_driver	
18h-1Ch	64	queue_device	

Table 316 - VIRTIO NET CONFIG Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
mac_47_16																															00h	
mac_15_0															status															04h		
max_virtqueue_pairs															mtu															08h		
																															0Ch	

Description as defined in virtio spec

Table 317 - VIRTIO NET CONFIG Fields

Offset	Bits	Name	Description
00h	31:0	mac_47_16	
04h	31:16	mac_15_0	
	15:0	status	

Table 317 - VIRTIO NET CONFIG Fields

Offset	Bits	Name	Description
08h	31:16	max_virtqueue_pairs	
	15:0	mtu	

8.27.20 VIRTIO BLK DEVICE EMULATION Object

Table 318 - VIRTIO BLK DEVICE EMULATION Object Layout

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										0h-04h
modify_field_select																														08h	
resources_on_emulation_manager																					enabled							vhca_id			
																														0Ch	
virtio_device (Table 312, “VIRTIO DEVICE Layout,” on page 584)																														10h-3Ch	
virtio_blk_config (Table 320, “VIRTIO BLK CONFIG Layout,” on page 589)																														40h-8Ch	

Table 318 - VIRTIO BLK DEVICE EMULATION Object Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										90h-ACh
virtio_q_configuration[0] (Table 314, “VIRTIO Q Configuration Layout,” on page 585)																															B0h-CCh
virtio_q_configuration[1] (Table 314, “VIRTIO Q Configuration Layout,” on page 585)																															D0h-...
...																															

Table 319 - VIRTIO BLK DEVICE EMULATION Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. 0x0: device_status
08h	31	resources_on_emulation_manager	If set, device will allocate resources on emulation manager instead of emulated function. This means that emulation resources must not be allocated on behalf by a vhca_tunnel. Supported only when HCA_CAP.resources_on_virtio_blk_emulation_manager==1.
	30	enabled	The field indicates whether the emulated function is enabled and ready to be used. Read Only.
	15:0	vhca_id	vhca identifier of the device emulation. Used only by device emulation manager (HCA_CAP.virtio_blk_device_emulation_manager==1).
10h-3Ch	384	virtio_device (Table 312, “VIRTIO DEVICE Layout,” on page 584)	See Table 312, “VIRTIO DEVICE Layout,” on page 584
40h-8Ch	640	virtio_blk_config (Table 320, “VIRTIO BLK CONFIG Layout,” on page 589)	See Table 320, “VIRTIO BLK CONFIG Layout,” on page 589

Table 319 - VIRTIO BLK DEVICE EMULATION Object Fields

Offset	Bits	Name	Description
90h- ...	256	virtio_q_configura- tion[...] (Table 314, “VIR- TIO Q Configuration Layout,” on page 585)	List of Virtio Q Configuration (See Table 314, “VIRTIO Q Configu- ration Layout,” on page 585). The array size is $1 + 2 \times \text{virtio\_blk\_device\_layout.max\_virt-}$ queue_pairs , where Queue index i corresponds to the <code>virtio_-</code> <code>queue_configuration[i]</code> .

Table 320 - VIRTIO BLK CONFIG Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												00h-04h
capacity																															00h-04h		
size_max																															08h		
seg_max																															0Ch		
																															10h		
cylinders															heads						sectors										14h		
																															18h		
blk_size																															1Ch		
																															20h		
physical_blk_exp								alignment_offset								min_io_size															24h		
opt_io_size																															28h		
																															2Ch		
writeback																															30h		

Table 320 - VIRTIO BLK CONFIG Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												34h
max_discard_sectors																													34h				
max_discard_seg																													38h				
discard_sector_alignment																													3Ch				
max_write_zeroes_sectors																													40h				
max_write_zeroes_segs																													44h				
write_zeroes_may_unmap																													48h				
																													4Ch				

Description as defined in virtio spec

Table 321 - VIRTIO BLK CONFIG Fields

Offset	Bits	Name	Description
00h-04h	64	capacity	
08h	31:0	size_max	
0Ch	31:0	seg_max	
14h	31:16	cylinders	
	15:8	heads	
	7:0	sectors	
1Ch	31:0	blk_size	
24h	31:24	physical_blk_exp	
	23:16	alignment_offset	
	15:0	min_io_size	
28h	31:0	opt_io_size	
30h	31:24	writeback	
34h	31:0	max_discard_sectors	
38h	31:0	max_discard_seg	

Table 321 - VIRTIO BLK CONFIG Fields

Offset	Bits	Name	Description
3Ch	31:0	discard_sector_align- ment	
40h	31:0	max_write_zeroes_sec- tors	
44h	31:0	max_write_zeroes_segs	
48h	31:24	write_zeroes_may_un- map	

8.27.21 VIRTIO_BLK_Q Object

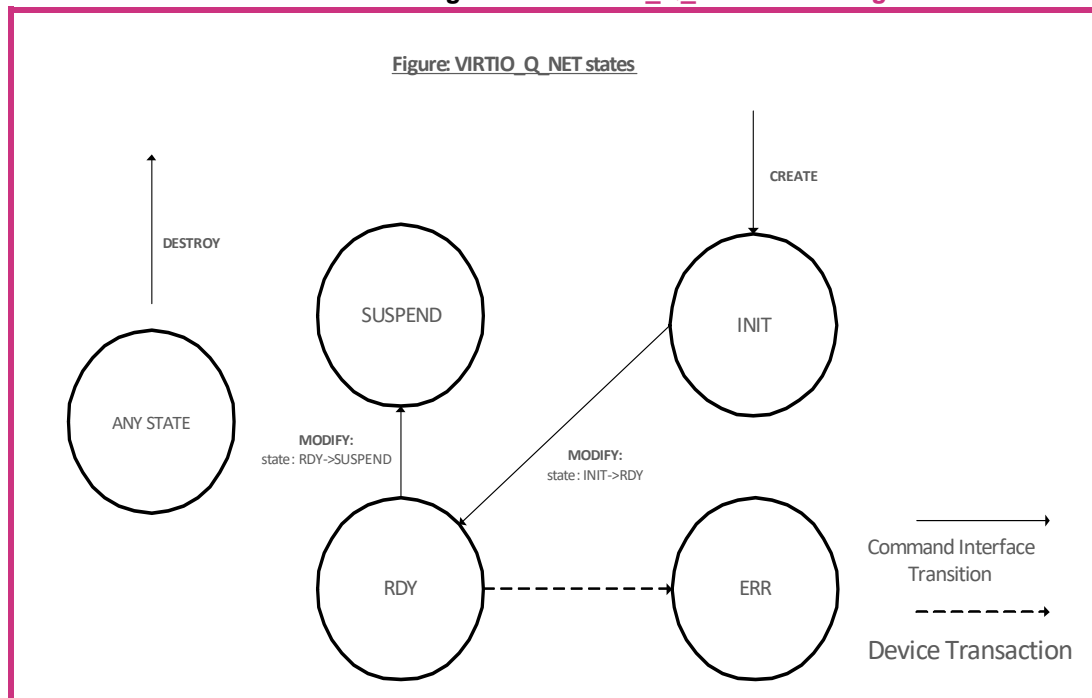
This object is supported only when `HCA_CAP.general_obj_types.VIRTQ_BLK_Q==1`.

The `VIRTIO_BLK_Q` object represents an emulated context for a `VIRTIO_BLK` Virtqueue which was created and managed by a `VIRTIO_BLK` driver as defined in `VIRTIO` Specification.

Software should provide all information of the `VIRTIO_BLK` Virtqueue it wants to accelerate when creating the context, and by modifying the state as described in the conditions below.

When the `VIRTIO_BLK_Q` is in operational state (RDY) the device is assumed to be the only entity which performs the `VIRTIO_BLK` Device-side data path engine, and communicates directly with the `VIRTIO_BLK` driver via `VIRTIO` doorbell kicks, MSI-X interrupts, updating and reading the used/available/descriptor rings, etc..as described in the `VIRTIO` specification.

Figure 45: VIRTIO_Q_BLK State Diagram



When creating a VIRTIO_BLK_Q object, SW should provide the required UMEM buffers needed internally to support the emulation. The number of umem buffers and their size are reported in VIRTIO_BLK_EMULATION_Capabilities.

After successful creation the VIRTQ object is in INIT state. SW then should modify the object and move it to RDY state.

SW may also modify the object's state to SUSPEND, which will result in stopping all I/O operations to be executed by the device. SW can then query the object for device information (mainly hw_available_idx and hw_used_idx) and it's guaranteed to have all information valid without any I/O operation pending by the device.

If modify object success (using MODIFY_GENERAL_OBJ), SW can assume that it also moved to the required state.

In case device encounter an error in a VIRTIO_BLK_Q operation, it moves the state to ERR and issues a corresponding Object Change Event. Then SW can query the object to get the detailed error as reported by syndrome.

For further information on how to use this object, please refer to Section 20.7, “VIRTIO Emulation Off-load,” on page 1008.

Table 322 - VIRTIO_BLK_Q Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset																				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-04h																				
modify_field_select																												00h-04h																								
																												08h																								
qpn_vhca_id																												0Ch																								
										dirty_bitmap_dump_enable	vhost_log_page															state				10h																						
											error_type																		qpn																							14h
											dirty_bitmap_mkey																												18h													
											dirty_bitmap_size																												1Ch													
dirty_bitmap_addr																												20h-24h																								
hw_available_index												hw_used_index																28h																								
																												2Ch-3Ch																								
virtio_q_context(Table 306, “VIRTIO_Q Format,” on page 575)																												40h-BCh																								

Table 323 - VIRTIO_BLK_Q Fields Description

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1means modifiable), and in MODIFY which fields to modify. Bit 0 : state Bit 3 : dirty_bitmap_params - For dirty_bitmap_mkey,dirty_bitmap_addr, dirty_bitmap_size and vhost_log_page Bit 4 : dirty_bitmap_dump_enable
0Ch	31:16	qpn_vhca_id	VHCA identifier where QP provided in qpn field is created. <u>Valid only when virtio_q.full_emulation is set.</u>
10h	21	dirty_bitmap_dump_enable	If set, the device will log the I/O transactions of a VIRTIO_BLK_Q to vHOST log dirty bitmap file. (used for live migration purposes). When set, dirty_bitmap_mkey, dirty_bitmap_size and dirty_bitmap_addr are valid.
	20:16	vhost_log_page	Log (base 2) of page size of the vHOST dirty bitmap file in granularity of 4KB.
	3:0	state	VIRTIO queue state 0x0: INIT 0x1: RDY 0x2: SUSPEND 0x3: ERR
14h	23:0	qpn	QP number to be pointed by this virtio Queue.
18h	31:0	dirty_bitmap_mkey	This mkey is used by the device to access the VHOST live migration log dirty bitmap file. Start address and size are indicated by dirty_bitmap_addr and dirty_bitmap_size Valid when dirty_bitmap_dump_enable==1.
1Ch	31:0	dirty_bitmap_size	Valid when dirty_bitmap_dump_enable==1.
20-24h	64	dirty_bitmap_addr	Valid when dirty_bitmap_dump_enable==1.
28h	31:16	hw_available_index	Initial value of the device emulation software available_idx to use for this VIRTIO_BLK_Q.
	15:0	hw_used_index	initial value of the device emulation software available_idx to use for this VIRTIO_BLK_Q.
40h-BCh	1024	virtio_q_context(Table 306, "VIRTIO_Q Format," on page 575)	Virtio Queue Context. See Table 306, "VIRTIO_Q Format," on page 575

8.27.22 HotPlug Device Object

Table 324 - HotPlug Device Object Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																													0h-04h			
																						initial_registers_valid	device_type				08h					
																						vhca_id										0Ch
device_pci_parameters (Table 326, “Device PCI Parameters Format,” on page 596)																													10h-1Ch			
																													20h-3Ch			
emulation_initial_registers																													40h-FCh			

Table 325 - HotPlug Device Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify. Bit 0: register_data
08h	8	initial_registers_valid	If set, initial registers structure is valid.
	7:0	device_type	Device Type 0x0: NVME 0x1: VIRTIO_NET 0x2: VIRTIO_BLK
0Ch	15:0	vhca_id	VHCA identifier that is allocated for the device. Read Only field.
10h-1Ch	128	device_pci_parameters (Table 326, "Device PCI Parameters Format," on page 596)	
40h-FCh	1536	emulation_initial_registers	Valid only when initial_registers_valid==1. See Table 328, "NVMe Initial Registers Format," on page 597 See Table 330, "Virtio Net Initial Registers Format," on page 597 See Table 332, "Virtio Blk Initial Registers Format," on page 598

Table 326 - Device PCI Parameters Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											
device_id																vendor_id																00h
revision_id								class_code																								04h
subsystem_id																subsystem_vendor_id																08h
																num_msix																0Ch

Table 327 - Device PCI Parameters Fields

Offset	Bits	Name	Description
00h	31:16	device_id	
	15:0	vendor_id	
04h	31:24	revision_id	
	23:0	class_code	
08h	31:16	subsystem_id	
	15:0	subsystem_vendor_id	
0Ch	15:0	num_msix	

Table 328 - NVMe Initial Registers Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h-BCh
nvme_registers																																

Table 329 - NVMe Initial Registers Fields

Offset	Bits	Name	Description
00h-BCh	1536	nvme_registers	

Table 330 - Virtio Net Initial Registers Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h-04h
device_features																															

Table 330 - Virtio Net Initial Registers Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											08h
																																00h-0Ch
																																10h-1Ch
																																20h-BCh

Table 331 - Virtio Net Initial Registers Fields

Offset	Bits	Name	Description
00h-04h	64	device_features	
08h	15:0	queue_size	
10h-1Ch	128	virtio_net_config(Table 316, "VIRTIO NET CONFIG Layout," on page 586)	See Table 316, "VIRTIO NET CONFIG Layout," on page 586

Table 332 - Virtio Blk Initial Registers Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset													
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	device_features																00h-04h						
												num_queues																queue_size																08h

Table 332 - Virtio Blk Initial Registers Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h-0Ch
virtio_blk_config (Table 320, “VIRTIO BLK CONFIG Layout,” on page 589)																											10h-5Ch				
																											60h-BCh				

Table 333 - Virtio Blk Initial Registers Fields

Offset	Bits	Name	Description
00h-04h	64	device_features	
08h	31:16	num_queues	
	15:0	queue_size	
10h-5Ch	640	virtio_blk_config(Table 320, “VIRTIO BLK CONFIG Layout,” on page 589)	See Table 320, “VIRTIO BLK CONFIG Layout,” on page 589

8.27.23 NVMF_BACKEND_CONTROLLER Object

Note this object replaces the CREATE_NVMF_BACKEND_CONTROLLER command to allow opening it by general object commands.

Table 334 - NVMF_BACKEND_CONTROLLER Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												0h-04h
modify_field_select																																0h-04h	
xrq_id																																08h	
initial_cqh_db_value																initial_sqt_db_value																0Ch	
state																								log_cmd_- timeout_us								10h	
doorbells_umem_id																																14h	
cq_umem_id																																18h	
sq_umem_id																																1Ch	
cqh_dbr_umem_offset																																20h-24h	
sqt_dbr_umem_offset																																28h-2Ch	
cq_umem_offset																																30h-34h	
sq_umem_offset																																38h-3Ch	
cq_size																																40h	
sq_size																																44h	

Table 334 - NVME_BACKEND_CONTROLLER Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									48h-7Ch

Table 335 - NVME_BACKEND_CONTROLLER Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY, which fields to modify.
08h	31:0	xrq_id	XRQ object ID
0Ch	31:16	initial_cqh_db_value	The initial completion queue head doorbell value.
	15:0	initial_sqt_db_value	The initial send queue tail doorbell value.
10h	31:24	state	0x0: READY 0x1: ERROR.
	4:0	log_cmd_timeout_us	Log(base2) of timeout (in useconds) for a command that is submitted to the NVMe backend device until command completion was seen. Device sends an error completion back to the initiator upon this timeout. An NVMe completion that arrives after the timeout will be ignored (and counted in comp_after_timeout counter). Must be between HCA_CAP.log_min_cmd_timeout and HCA_CAP.log_max_cmd_timeout. 0 means that the device uses its default timeout value.
14h	31:0	doorbells_umem_id	A UMEM object ID that maps the backend controller doorbells area. Must be aligned to 4 Bytes.
18h	31:0	cq_umem_id	A UMEM object ID that is mapped to the backend controller CQ. Must be aligned to 64 Bytes.
1Ch	31:0	sq_umem_id	A UMEM object ID that is mapped to the backend controller SQ. Must be aligned to 64 Bytes.
20h-24h	64	cqh_dbr_umem_offset	Offset within the doorbells UMEM object of the CQ head doorbell. Must be aligned to 4 Bytes.
28h-2Ch	64	sqt_dbr_umem_offset	Offset within the doorbells UMEM object of the SQ tail doorbell. Must be aligned to 4 Bytes.

Table 335 - NVMF_BACKEND_CONTROLLER Object Fields

Offset	Bits	Name	Description
30h-34h	64	cq_umem_offset	Offset within the CQ UMEM where the CQ begins. Must be aligned to 64 Bytes.
38h-3Ch	64	sq_umem_offset	Offset within the SQ UMEM where the SQ begins. Must be aligned to 64 Bytes.
40h	31:0	cq_size	
44h	31:0	sq_size	

8.27.24 NVMF_NAMESPACE Object

Note this object replaces the ATTACH_NVMF_NAMESAPCE command to allow opening it by general object commands.

Table 336 - NVMF_NAMESPACE Object Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																															0h-04h	
xrq_id																															08h	
frontend_namespace																															0Ch	
backend_namespace																															10h	
backend_controller_id																															14h	
lba_data_size															metadata_size																18h	
nvmf_namespace_counter_set_id																															1Ch	
																															20h-3Ch	

Table 337 - NVMF_NAMESPACE Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
08h	31:0	xrq_id	XRQ object ID
0Ch	31:0	frontend_namespace	Front-end namespace to be attached to XRQ.
10h	31:0	backend_namespace	NVMe backend namespace to be used to handle the attached front-end namespace.
14h	31:0	backend_controller_id	NVMF_BACKEND_CONTROLLER object ID, instead of 16 bit seq number
18h	31:16	lba_data_size	LBA data size in granularity of Bytes. Minimal allowed 512B. Must be power of 2.
	15:0	metadata_size	The number of metadata bytes per LBA. 0 means no metadata. The maximum support size is HCA_CAP.max_lba_metadata_size.
1Ch	31:0	nvmf_namespace_counter_set_id	Object ID for NVMF namespace counters set. 0 mean no counter set.

8.27.25 NVMF_NAMESPACE_COUNTER_SET Object

This object is used to create a counters set which can be connected to an NVMF_NAMESPACE object to count operations on it. Multiple NVMF_NAMESPACE objects may use the same counter set. To clear counters, the software can use a MODIFY command with value 0 on the counters while setting the modify_field_select.

Table 338 - NVMF_NAMESPACE_COUNTER_SET Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										0h-04h
modify_field_select																															08h-0Ch

Table 338 - NVME_NAMESPACE_COUNTER_SET Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											10h-14h
num_read_cmd																												10h-14h				
num_read_blocks																												18h-1Ch				
num_write_cmd																												20h-24h				
num_write_blocks																												28h-2Ch				
num_write_inline_cmd																												30h-34h				
num_flush_cmd																												38h-3Ch				
num_error_cmd																												40h-44h				
num_backend_error_cmd																												48h-4Ch				
																												50h-7Ch				

Table 339 - NVME_NAMESPACE_COUNTER_SET Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
10h-14h	64	num_read_cmd	Number of Read Commands. Note that only the 32 LSB bits are valid.

Table 339 - NVME_NAMESPACE_COUNTER_SET Object Fields

Offset	Bits	Name	Description
18h-1Ch	64	num_read_blocks	Number of Read Blocks. Note that only the 32 LSB bits are valid.
20h-24h	64	num_write_cmd	Number of Write Commands. Note that only the 32 LSB bits are valid.
28h-2Ch	64	num_write_blocks	Number of written Blocks. Note that only the 32 LSB bits are valid.
30h-34h	64	num_write_inline_cmd	Number of Write Commands with inline. Note that only the 32 LSB bits are valid.
38h-3Ch	64	num_flush_cmd	Number of Flush Commands. Note that only the 32 LSB bits are valid.
40h-44h	64	num_error_cmd	Number of Error Commands.(Bad or unsupported) Note that only the 32 LSB bits are valid.
48h-4Ch	64	num_backend_error_cmd	Number of commands that got bad status completion from the backend NVMe Note that only the 32 LSB bits are valid.

8.27.26 VHCA Tunnel Object

An Emulation Manager that is allowed to manage emulated functions (such as NVMe) needs to perform commands on behalf of these functions. To allow that, the emulation manager needs first to create a commands tunnel on this emulated function by creating VHCA_TUNNEL Object.

The tunnel identifier (16 LSB of it) is set when performing a command on behalf of an emulated function to identify on which function this command is to be performed.

Each emulated function can be managed only by a single emulation manager, using a single vHCA Tunnel object. I.e. only a single tunnel object is allowed to be opened on a vhca_id.

Note that destroying this object causes internal resources reset on the emulated function.

Table 340 - VHCA_TUNNEL Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												0h-04h
modify_field_select																																	

Table 340 - VHCA_TUNNEL Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										08h
																															0Ch-3Ch

Table 341 - VHCA_TUNNEL Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
08h	15:0	vhca_id	VHCA identifier of the device emulation that the tunnel is opened.

8.27.27IPSEC Offload Object

The IPsec Offload Object is the basic underlying structure of an IPsec flow. It contains all parameters required to manage ESP sequence number, perform replay protection, encrypt/decrypt and authenticate packets.

This object should be used as an action in a Flow Table Entry to perform the IPsec operation on packets matching the Flow. When the IPsec decryption operation is completed, ipsec_syndrom will be available for matching to decide on the following actions according to the success or failure of the operation.

Table 342 - IPSEC_OFFLOAD Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										0h-04h

Table 342 - IPSEC_OFFLOAD Object Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											08h
full_offload		esn_overlap				icv_length						aso_re- turn_reg																				
esn_msb																												0Ch				
							dekn																					10h				
salt																												14h				
implicit_iv																												18h-1Ch				
																												20h-3Ch				
ipsec_aso (See Table 344, “IPSEC_ASO Format,” on page 609)																												40h-7Ch				

Table 343 - IPSEC_OFFLOAD Object Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY, which fields to modify. Bit 0: esn_overlap Bit 1: esn_msb

Table 343 - IPSEC_OFFLOAD Object Fields

Offset	Bits	Name	Description
08h	31	full_offload	IPsec offload type 0x0: CRYPTO_OFFLOAD 0x1: FULL_OFFLOAD
	29	esn_en	When set, ESN is enabled
	28	esn_overlap	New/Old indication of the High sequence number MSB. 0x0: OLD 0x1: NEW [Internal] TBD need description of how to update
	25:24	icv_length	Authentication Tag length 0x0: LEN_816B 0x1: LEN_12B 0x2: LEN_468B other values are reserved
	19:16	aso_return_reg	Indicates which Register C will hold the IPSEC ASO result 0x0: REG_C_0_1- Syndrome will be set in register C 0[7:0] and ESP SN will be set in register C 1 0x1: REG_C_2_3 - Syndrome will be set in register C 2[7:0] and ESP SN will be set in register C 3 0x2: REG_C_4_5 - Syndrome will be set in register C 4[7:0] and ESP SN will be set in register C 5 0x3: REG_C_6_7 - Syndrome will be set in register C 6[7:0] and ESP SN will be set in register C 7 other values are reserved. Value must comply with HCA_CAP.flow_meter_reg_id. Valid only if full_offload is set
0Ch	31:0	esn_msb	MSB of the Extended Sequence Number (ESN)
10h	23:0	dekn	Data Encryption Key (DEK) number
14h	31:0	salt	See implicit_iv
18h-1Ch	64	implicit_iv	The IV is inserted into the GCM engine is calculated by: GCM_IV[95:0] = {salt, implicit_iv XOR original packet.IV}. The packet's IV field is replaced with implicit_iv XOR original packet.IV. [Internal] TBD what does this mean: This field must be zero for Inbound packets.
40h-7Ch	512	ipsec_aso (See Table 344, "IPSEC_ASO Format," on page 609)	Advanced Steering Operation (ASO) context for Full IPsec offload. valid only if full_offload is set. See (Table 344, "IPSEC_ASO Format," on page 609).

Table 344 - IPSEC_ASO Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2																			00h			
valid																															
mode																															
window_size																															
soft_lifetime_arm																															
hard_lifetime_arm																															
remove_flow_enable																															
esn_event_arm																															

Table 345 - IPSEC_ASO Fields

Offset	Bits	Name	Description
00h	31	valid	Must be set to 0x1
	29:28	mode	Mode for optional operations 0x0: NO_OPTIONAL_OP 0x1: REPLAY_PROTECTION - packets are checked vs. the replay protection window. Packets newer than window high bound advance window. Packets in window are checked against replay. Valid for Decryption operation only. 0x2: INCREMENT_IPSEC_SN - Increment IPsec Sequence Number. Valid for Encryption operation only. other values are reserved
	27:26	window_size	Replay Protection Window size. Valid only when mode is set to REPLAY_PROTECTION 0x0: WIN_32BIT 0x1: WIN_64BIT 0x2: WIN_128BIT 0x3: WIN_256BIT
	25	soft_lifetime_arm	Write 0x1 to arm the device to generate an event when soft-lifetime threshold is reached. Once the event is fired, the event must be re-armed (set this field to 0x1) to trigger a new event for this IPsec context
	24	hard_lifetime_arm	Write 0x1 to arm the device to generate an event when hard-lifetime threshold is reached. Once the event is fired, the event must be re-armed (set this field to 0x1) to trigger a new event for this IPsec context.
	23	remove_flow_enable	when set, remove flow is enabled
	22	esn_event_arm	Write 0x1 to arm the device to generate an event when ESN wrap-around is reached. Once the event is fired, the event must be re-armed (set this field to 0x1) to trigger a new event for this IPsec context.
04h	31:0	remove_flow_packet_count	Packet counter, Decrements for every packet passing through the operation. 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
08h	31:0	remove_flow_soft_lifetime	Soft Lifetime threshold value. When remove_flow_packet_count reaches this value a soft lifetime event is triggered (if armed)
1Ch	31:0	mode_parameter	When mode is set to REPLAY_PROTECTION, reflects the LSB of the replay protection high bound When mode is set to INCREMENT_IPSEC_SN, reflects the current ESP header sequence number (same number which is used in packet)

Table 345 - IPSEC_ASO Fields

Offset	Bits	Name	Description
20h-3Ch	256	replay_protection_window	Replay Protection Window. valid only When mode is set to REPLAY_PROTECTION

8.27.28 Match Definer Object

This object is supported when `HCA_CAP.general_obj_types[MATCH_DEFINDER] == 1`.

The maximum supported number of match definers is reported in `HCA_CAP.max_num_match_definer`.

Table 346 - MATCH_DEFINDER Object Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																															00h-04h	
																															08h-0Ch	
																																10h
																															14h-3Ch	
match_mask																															40h-7Ch	

Table 347 - MATCH_DEFINDER Fields

Offset	Bits	Name	Description
00h-04h	64	modify_field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.
10h	15:0	format_id	Identifier of one of the supported Match definer formats. The supported formats are reported in <code>HCA_CAP.match_definer_format_supported</code> . 0x3F: MATCH_DEFINDER_FORMAT_63 - See Table 348, “Match_Definer_Format_63 Layout,” on page 612

Table 347 - MATCH_DEFINER Fields

Offset	Bits	Name	Description
40h-7Ch	512	match_mask	bitmask indicating which bit of the fields defined by the format will be matched. Cleared bits indicate that the appropriate bit is not matched to the packet. for formats smaller than 64B, the higher offsets of the match mask are reserved, i.e. match_mask={mask, reserved}

The formats are defined below:

Table 348 - Match_Definer_Format_63 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
outer_ipv4_dip																																10h
outer_ipv4_sip																																14h
outer_udp_tcp_sport																outer_udp_tcp_dport																18h
						outer_frag	outer_tcp_flags									outer_dscp					outer_ecn	outer_protocol						1Ch				
																																20h-3Ch

Table 349 - Match_Definer_Format_63 Fields

Offset	Bits	Name	Description
10h	31:0	outer_ipv4_dip	
14h	31:0	outer_ipv4_sip	
18h	31:16	outer_udp_tcp_sport	
	15:0	outer_udp_tcp_dport	

Table 349 - Match_Definer_Format_63 Fields

Offset	Bits	Name	Description
1Ch	25	outer_frag	
	24:16	outer_tcp_flags	
	15:10	outer_dscp	
	9:8	outer_ecn	
	7:0	outer_protocol	

8.28 Command Interface

The HCA command interface is used for:

- configuring the HCA
- the handshake between hardware and system software
- handling (querying, configuring, modifying) HCA objects

The HCA is configured using the command queues. Each function has its own command queues to get commands from its HCA driver. A detailed reference for the syntax of the command interface commands is provided in Chapter 29, “Command Reference” (page 1647).

8.28.1 HCA Command Queue

The command queue is the transport that is used to pass commands to the HCA.

The command queue is a buffer on fixed-size entries. Each entry can contain one command queue entry. The number of entries and the entry stride can be retrieved from the initialization segment of the HCA Bar. See Table 14, “Initialization Segment”.

The command queue resides in physically contiguous 4KB memory chunk aligned on its size. The physical address of this memory chunk is configured by SW to the initialization segment.

The command queue is a directional way to send commands. SW will put a command request in it and will receive a command response. When posting an entry to the command queue, SW will ring the command DoorBell vector in the initialization segment by writing to this vector with the respective (to the newly posted command) bit set. If EQ is set and mapped to this command queue, SW will be informed on command completion by EQE.

SW can execute a single write to the DoorBell vector for multiple commands by setting multiple bits in the DoorBell vector.

Hardware Function may execute and complete the commands in an out-of-order manner. Once a command is completed, the ownership bit of the command is updated and the event is generated.

It is SW's responsibility to ensure successful generation of this event. Consequently, if no room is left in EQ buffer, the command should not be sent until the EQ consumer index has been updated (using EQ doorbell).

The event data includes the command completion vector. Bits in this vector are set for commands completed after the former event.

As an alternative to waiting for events, SW can poll the ownership bit in the command to understand when the command is completed.

Before EQ was generated, SW cannot wait for events to understand the status of commands. Once EQ is created and mapped, as well as mapped for command completion events on the command queue, SW can wait for events.

Note that the CREATE_EQ command that maps EQ to the command queue must be posted to the command queue while it is empty and no other command should be posted.

Table 350, "Command Queue Entry Layout," on page 614 shows the layout of the command. Note that status bit and all reserved fields should be set to zero by SW when posting the command.

When a command request is carried by a command queue entry, the first 16 bytes of the command reside as inline data in the command_input_inline_data field. The rest of the command will be in the input mailbox. For the command response, the first 16 bytes will reside in the command_output_inline_data. The rest of the command response will be in the output mailbox.

Table 350 - Command Queue Entry Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type																																00h
input_length																												04h				
input_mailbox_pointer[63:32]																												08h				
input_mailbox_pointer[31:9]																												0Ch				
command_input_inline_data																												10h-1Ch				
command_output_inline_data																												20h-2Ch				
output_mailbox_pointer[63:32]																												30h				
output_mailbox_pointer[31:9]																												34h				
output_length																												38h				
token								signature														status				ownership	3Ch					

Table 351 - Command Queue Entry Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	type	Type of transport that carries the command 0x7: PCIe_cmd_if_transport	
04h	31:0	input_length	Input command length in bytes	
08h	31:0	input_mailbox- _pointer[63: 32]	Pointer to input mailbox	
0Ch	31:9	input_mailbox- _pointer[31: 9]		
10h- 1Ch	128	com- mand_in- put_inline_d ata	The first 16 bytes of the command input	
20h- 2Ch	128	com- mand_out- put_inline_d ata	The first 16 bytes of the command output	
30h	31:0	output_- mailbox- _pointer[63: 32]	Pointer to output mailbox	
34h	31:9	output_- mailbox- _pointer[31: 9]		
38h	31:0	out- put_length	Output command length in bytes	

Table 351 - Command Queue Entry Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
3Ch	31:24	token	Token of the command. Should have the same value in the command and the mailbox blocks.	
	23:16	signature	8 bit signature of the command queue entry, Does not include the mailbox.	
	7:1	status	Command delivery status. SW should set this field to 0 when posting the command. 0x0: SUCCESS - no errors 0x1: SIGNATURE_ERR 0x2: TOKEN_ERR 0x3: BAD_BLOCK_NUMBER 0x4: BAD_OUTPUT_POINTER - pointer not aligned to mailbox size 0x5: BAD_INPUT_POINTER- pointer not aligned to mailbox size 0x6: INTERNAL_ERR 0x7: INPUT_LEN_ERR- input length less than 0x8 0x8: OUTPUT_LEN_ERR- output length less than 0x8 0x9: RESERVED_NOT_ZERO 0x10: BAD_COMMAND_TYPE	
	0	ownership	SW should set to 1 when posting the command. HW will change to zero to move ownership bit to SW.	

For commands posting, notification on completion, and events generation see Section 7.1, “Initialization,” on page 243.

8.28.1.1 Command Timeout

The device is not setting timeout to the commands. The driver can start a timeout when posting a command. It is not clear how to set this value. TBD.

8.28.1.2 Mailbox Format

Input and output mailboxes are shown in Table 352, “Command Interface Mailbox Layout,” on page 616. If the mailbox data is larger than the size of the mailbox-data field, the first part of the data will be in the first mailbox block, the second part in the second block etc.

Table 352 - Command Interface Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
mailbox_data																																00h-1FCh

Table 352 - Command Interface Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															200h-22Ch	
next_pointer[63:32]																															230h	
next_pointer[31:10]																															234h	
block_number																															238h	
									token									ctrl_signature									signature					23Ch

Table 353 - Command Interface Mailbox Field Descriptions

Offset	Bits	Name	Description	Access
00h-1FCh	4096	mailbox_data	Data in the mailbox	
230h	31:0	next_pointer[63:32]	Pointer to the next mailbox page (if needed). If no additional block is needed, the pointer should be 0.	
234h	31:10	next_pointer[31:10]	Pointer to the next mailbox page (if needed). If no additional block is needed, the pointer should be 0.	
238h	31:0	block_number	Sequence number of the block. Starting by 0 and increment for each block on the linked list of blocks.	
23Ch	23:16	token	Token of the command. Should have the same value in the command and the mailbox blocks.	
	15:8	ctrl_signature	Signature dwords 1D0h-1FCh result of bitwise XOR of all those DWORDs including the ctrl_signature should result in 0xFF (while the signature byte is considered as 0).	
	7:0	signature	Signature of the current mailbox page. bitwise XOR of all bytes of the page, including the reserved fields and the signature fields should result in 0xFF.	

8.28.1.2.1 Calculating Mailbox Signatures

Mailbox signatures are calculated for each mailbox block independently. Calculation is done in the following way:

- update all fields of the mailbox block with their values

- write '0' to the ctrl_signature and the signature fields
- calculate bitwise-xor of the ctrl part (bytes 1C0h - 1FFh)
- invert the result and write it to the ctrl_signature field
- re-calculate bitwise-xor of the whole mailbox block
- invert the result and write it to the signature field
- before posting of command, the signatures of the command queue entry and the input mailbox should be calculated. In addition, the ctrl_signature of the output mailbox blocks should be calculated. This is to protect the output mailbox pointers and token fields.

8.28.1.3 Command Data Layout

The layout of the input command is shown in Table 354, “Command Input Data Layout,” on page 618.

Table 354 - Command Input Data Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
opcode																																
																op_mod																
command[0]																														08h		
command[1]																														0Ch		
...																														...		

Table 355 - Command Input Data Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod	Opcode modifier, if there is no description, then op_mod must be 0.	
08h-...	31:0	command[...]		

The layout of the output command is shown in Table 356, “Command Output Data Layout,” on page 619.

Table 356 - Command Output Data Layout

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																														004h		
command_output																														08h		

Table 357 - Command Output Data Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome	Syndrome on the command. Valid only if status = 0x0.	
08h	31:0	command_output	Command output. Valid only if status = 0x0.	

8.29 ICMD Interface

ICMD interface is a method allowing a SW tool to send commands to the NIC/HCA. The interface is intended to allow the tools to operate in parallel and independently of the host software.

ICMDs are issued through a Gateway implemented over a Vendor Specific PCI Capability. See Table 9, "Vendor Specific Capability Layout", using the ICMD *space* access.

Some of the ICMDs require an input mailbox for holding additional information used for the command execution, and/or an output mailbox for holding results of the command execution.

Table 358, "ICMD Configuration Domain" describes the location of the different elements required for ICMD execution in the ICMD configuration domain.

Table 358 - ICMD Configuration Domain

Name	Address	Access	Content
ICMD Control	0x0	R/W	See Section 8.29.1, "ICMD Control"
External Mailbox Mkey	0x8	R/W	see Section 8.25.2.1, "External Mailbox"
Mailbox Size	0x1000	RO	Maximal size in bytes of the ICMD mailbox
Tools Flags	0x1004	R/W	

Table 358 - ICMD Configuration Domain

Name	Address	Access	Content
Syndrome	0x1008	RO	Advanced error reporting for a failed ICMD command. Valid when the command status indicates an error.
Mailbox	0x100000	R/W	Starting point of the data structure holding the input parameters and the output parameters of the command.

8.29.1 ICMD Control

The control segment of the ICMD interface is used to initiate a command, to monitor the command's completion and to verify the completion status.

Table 359 - ICMD Control Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																status												exmb	busy	00h		

Table 360 - ICMD Control Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	Command opcode. See Table 2731, "ICMD Commands List Sorted by Opcode," on page 2267
	15:8	status	Command delivery status. Valid when busy field is 0x0. 0x0 - no error 0x1 - invalid opcode 0x2 - invalid command TBD what is this ? 0x3 - operational error TBD what is this ? 0x4 - bad input parameter 0x5 - busy TBD what is this ? 0x6 - ICM not available TBD what is this ? 0x7 - write protection error TBD what is this ? Other values are reserved for future error codes and should be handled as if the command failed for an unknown reason.
	1	exmb	External Mailbox. When set, the ICMD uses a memory buffer as a mailbox instead of the normal ICMD mailbox.
	0	busy	Set by the command driver to initiate the command. Cleared by the device when the command completes.

8.29.2 Using ICMD Commands

Issuing an ICMD command should adhere to the following flow:

1. Clear the Mailbox according to the **Mailbox Size**.

2. Set the input Mailbox for the command (if defined).
3. Set the command Control - set the *opcode*, clear the *status* and set the *busy* field.
4. Poll the command Control - wait for *busy* field to be cleared.
5. Check for correct completion of the command by verifying the status is “no errors”. If the command failed, the *syndrome* field may be queried to provide additional information about the reason for the command failure.
6. Read the output Mailbox for the command (if defined).

Note: The Gateway ownership should be acquired and held through the entire execution of the command to avoid conflicts with other users of the interface.

8.29.2.1 External Mailbox

When the agent activating the ICMD has access to a registered memory buffer, it may use it as a replacement for the ICMD mailbox. Using an easily accessible buffer may improve ICMD execution performance, especially for ICMDs that require large mailboxes.

The support for an external mailbox is indicated by *icmd_exmb* field in “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities”. The MKey defining the external mailbox should be set in the *External Mailbox MKey* field in Table 358, “ICMD Configuration Domain”, after the VSC Gateway ownership is taken. To indicate that an ICMD should be executed using the external mailbox, the *exmb* field should be set in Table 359, “ICMD Control Layout”.

Note: Once the Gateway ownership is released, a previously set MKey is considered invalid, and must be set again after ownership is reacquired.

8.29.3 ICMD Command Formats

The ICMD commands supported by the device are described in [Chapter 30, “ICMD Command Reference”](#) on page 2267 along with their opcodes and mailbox formats.

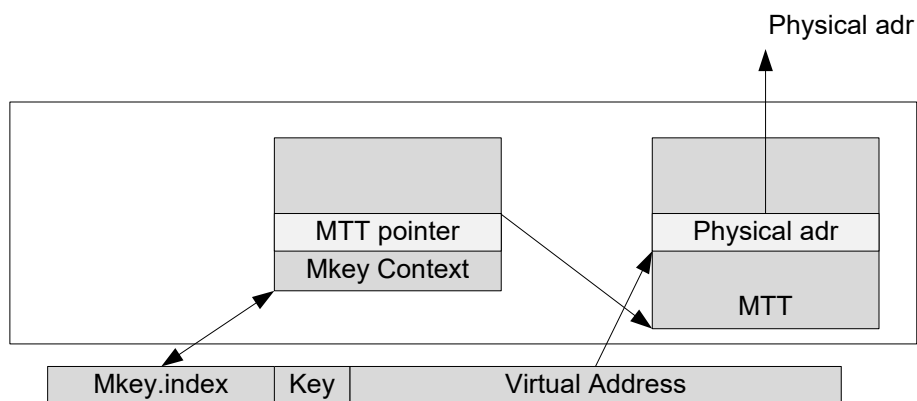
8.29.4 ICMD Capabilities

A SW tool may inquire about the capabilities supported by the ICMD interface. This can be achieved by the “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities” command over the ICMD interface. If this command returns an *invalid opcode* status, all capabilities are implicitly unsupported.

9 Address Translation and Protection

Each memory request generated by the HCA undergoes address translation and protection checks as illustrated in [Figure 46](#).

Figure 46: Address Translation Stages

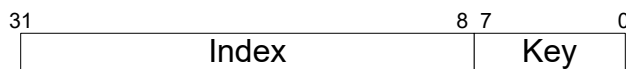


The device translates the given virtual address to a physical address using the given memory key as an address space identifier. Memory protection checks are done in this stage.

9.1 Virtual to Physical Address Translation and Protection Checks

Each memory request specifies a virtual address space identifier (Memory Key or MKey) and a virtual address within this address space. An MKey is a 32-bit address space identifier consisting of two fields - index and key. The Index field identifies the MKey. The key field can be altered by the application program for various reasons such as a generation identifier when an MKey is reused. The logical structure of an MKey is shown in [Figure 47](#).

Figure 47: MKey Structure



Address translation performs the bulk of the work. A memory region context is retrieved using the MKey given in the memory operation. Internally, MKeys are managed in the Memory Protection Table (MPT).

Once a memory region context is located, the memory operation is validated against the parameters in the memory region context:

1. Memory key domain – The protection domain for the region specified by MKey is validated against the protection domain in the QP context.
2. Boundary check – The virtual address and length of access are validated against the region/window boundaries.

3. Operation check – Operation (read/write/atomic) and location (local/remote) are validated against the memory region/window rights.

If all the previous validations passed, then the virtual address is translated into a physical address.

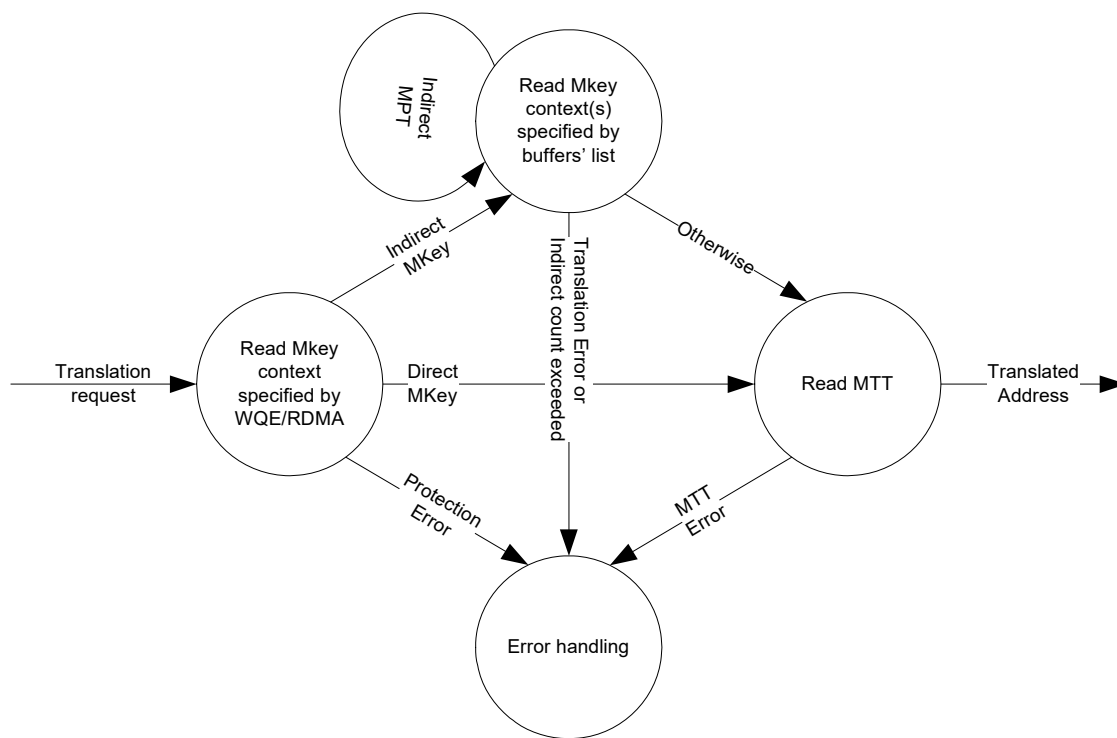
The memory region's base address is subtracted from the Virtual Address (VA) given in the operation, giving an offset into the memory region. With this offset, one of the memory translation table entries (MTT) associated with the memory key context can be selected. The type of memory region will determine what the MTT points to and hence how to pick the right one. There are 2 different types of memory regions:

- *Direct mode:* Each MTT entry translates a naturally-aligned page. The page size is specified in the MKey context; HW supports page sizes from 4KB to 2 GB. The page size must be a power of two. In this mode, a physical address may now be calculated. The offset from the previous step is divided by the size of each MTT entry, giving the correct MTT entry. The remainder of this division is added to the physical address indicated in the MTT entry giving the physical address. If necessary, spanning multiple MTT entries is done automatically using the same process.
- *Indirect mode:* Each KLM entry specifies another memory key, an address within the address space specified by this memory key and the length in bytes of the segment represented by this KLM entry. The KLM entries for the memory region are “walked” until the correct entry for the given offset is found. The remainder of the offset is added to the address in the KLM entry and the entire process is repeated. HW protects itself from the possibility of infinite recursion. Note that memory protection checks must pass on ALL indirect MKeys as well as the direct MKey, or the operation will fail. For executing memory operation, the appropriate access-right (for example, local write) should be in all related MKeys. Indirect memory regions allow the combination of multiple sparse memory buffers to a single virtually-contiguous memory buffer that can be used for I/O operation. See [“Address Translation Indirection” on page 624](#) for details.

In the case where the KLM buffers are associated with a fixed memory buffer size, it is recommended to use Fixed Size KLM access mode.

The state machine flow of address translation is shown in [Figure 48](#).

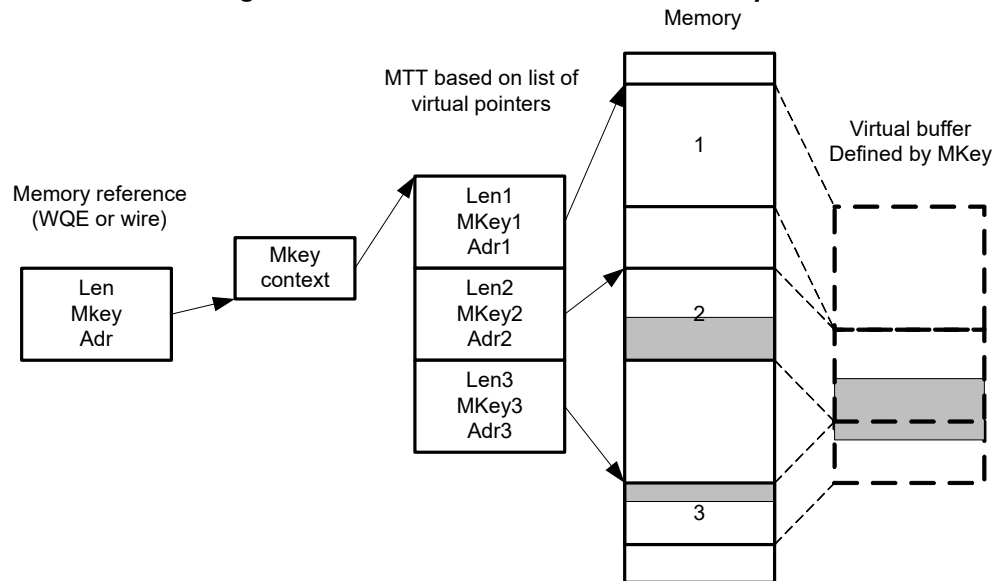
Figure 48: Address Translation State Machine



9.1.1 Address Translation Indirection

The device has extended memory management functionality that enables specifying an arbitrary list of memory buffers (scatter/gather list) as a virtual address space. The memory translation table (MTT) entries associated with a given memory key can be a list of buffer pointers in a structure similar to the scatter/gather list used in WQEs. When used this way, the MKey is called an indirect MKey. The size and alignment of each individual buffer (MTT entry) are not constrained. When an indirect MKey is used, the address given in the memory reference specifies an address within the virtual address space defined by the list of memory buffers (MTT entries) of the MKey. In turn, the given address identifies an individual pointer(s) to be used for the next step in the address translation process. The next step in address translation uses the memkey and virtual address of the pointer identified in the first step. See [Section 9.1, “Virtual to Physical Address Translation and Protection Checks,” on page 622](#) for a detailed description of address translation calculation. A simple case of one-step indirect address translation flow is illustrated in [Figure 49](#).

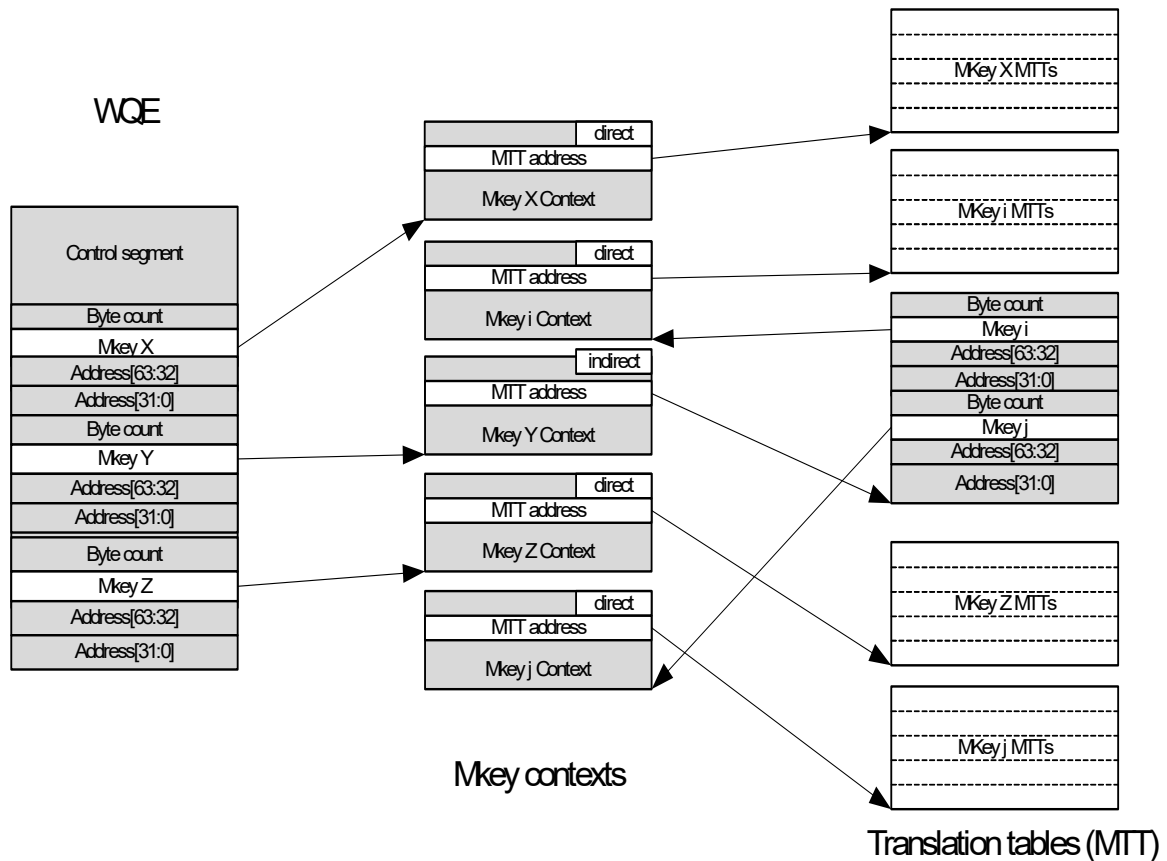
Figure 49: Address Translation Flow Example



The original memory reference $\{mkey, len, adr\}$ accesses an indirect memkey, as specified in the *MKey* context. This *MKey*'s MTT entries contain a list of buffers forming a memory region specified by *MKey*. The *adr* operand of the original memory reference is used to calculate an offset from the beginning of this region. HW scans the pointers identifying which one should be used to start the memory access specified by the original reference. In the example in [Figure 49](#), the second pointer is identified as the one to be used as the first byte address to be used. Subsequently, the *len* field of the original memory reference is used to figure out which pointers will be used to complete the memory request. In this example the memory access starts at the middle of the buffer specified by the second pointer and carries over into the third one.

A single Work Request (WQE) can mix memkeys of different types. [Figure 50](#) illustrates an example of a WQE using both direct and indirect address translations for different scatter/gather entries.

Figure 50: Address Translation Indirection in WQE



The indirection capability enables a host to specify a scatter/gather list of unconstrained size and length memory buffers as the target of RDMA operations using a very fast lightweight memory registration process. Nesting of indirect translations is limited to the number of iterations that is globally configured in the device. If the number of translation indirections exceeds the configured limit, the address translation operation terminates with an error.

Indirect MKeys are programmed via the UMR Work Request, for details please see [“User-Mode Memory Registration \(UMR\)” on page 918](#).

9.1.2 On-demand Paging Support

The HCA supports on-demand paging (ODP) of host memory for data buffer access. This functionality is indicated by the **pg** bit in the device capabilities reported by the HCA via the QUERY_HCA_CAP command. When this bit is set, ODP-enabled memory regions may be created by the CREATE_MKEY command. The resulting MKey can be enabled for address translation (i.e., free=0 in the MKey context – [Table 362, “MKey Context Format,” on page 629](#)), even if the associated memory translation table (MTT) contains non-accessible entries. The specific transport operations that support ODP may be obtained by querying the ODP capabilities using QUERY_HCA_CAP with a special **opc_mod** (see [Table 1568, “QUERY_HCA_CAP Input Structure](#)

Field Descriptions,” on page 1667). Unsupported operations that incur page faults result in “Local Protection Error” completions, “Remote Access Error” NAK syndromes, and “Local Access Violation” affiliated asynchronous events as specified in Table 361.

Table 361 - Actions if ODP is Not Supported

Operation	Action if ODP is Not Supported
Requester Send, RDMA, and atomic WQEs	Generate a “Local Protection Error” completion
Responder Receive WQEs	Generate a “Local Protection Error” completion and send a “Remote Operational Error” NAK for reliable services as specified in “Responder Class A Fault Behavior” in the IB specification.
Responder handling of incoming RDMA and atomic requests	Handle an R_Key violation as specified in “Responder Error Behavior Summary” in the IB specification. That is, raise a “Local Access Violation” affiliated asynchronous event and send a “Remote Access Violation” NAK for reliable services.

Two bits, *rd_en* and *wr_en*, in the memory translation table (see Table 365, “Memory Translation Table (MTT) Entry Layout,” on page 634) entry indicate the paging status of each memory page. If on-demand paging is supported, and, during a memory access, these bits indicate that a page is not present (i.e. both bits are clear) or its access permissions conflict with the operation. The HCA generates an event (see Section 8.20.10.1, “Page Fault Event,” on page 483) reporting the memory key, the virtual address and the QP number that attempted to access the page. The driver fixes the MTTs and updates the HCA with a PAGE_FAULT_RESUME (see Section 29.6.4, “PAGE_FAULT_RESUME,” on page 1869).

If a page fault occurs during a send operation, the packet experiencing the page fault is not sent to the network and the Send operation is suspended by the internal scheduler. The page is swapped in and the operation is resumed.

If a page fault occurs during a receive operation, the packet causing the exception is dropped, and recovery is done by the transport layer. If the RQ implements reliable transport, an RNR NAK message is sent to the requester. Once the page is swapped in, a retransmission by the sender continues the communication. If the RQ experiencing a page fault deploys a SW transport, the page fault has the same effect as a loss of a packet in the fabric and the SW layer recovers.

When using SRQ in conjunction with ODP, the HCA might generate error CQEs with remote abort syndrome for WQEs in the SRQ. These syndromes signal that the HCA was unable to use the WQE, and it should be reposted. The application should also attempt to ensure that the content pointed to by the work request posted is available in the HCA page tables. If the entire WQE is not available, it will be rejected with the remote abort syndrome. The WQE might be rejected with remote abort syndrome also due to other, micro-architectural reasons. In any case, the recommended application flow is to repost the WQE after ensuring that all of the memory pages referenced to by the WQE are available.

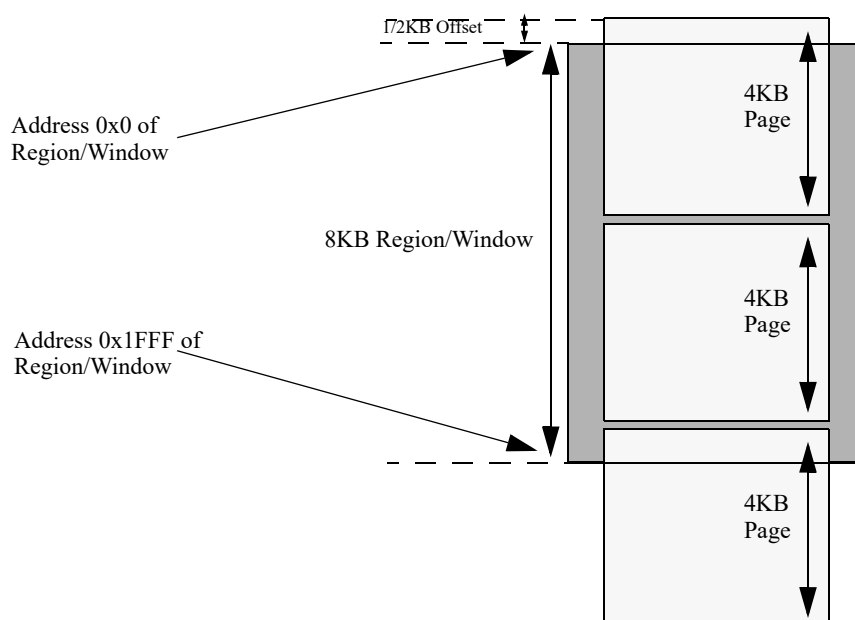
9.2 Zero-Based Virtual Address Regions and Windows

Zero-based virtual address regions are regions that start with virtual address zero. This property of the region is specified in the MKey context. Memory Windows cannot be bound onto zero-based regions.

The start of a zero-based memory region does not need to be aligned with the first page of the region (see [Figure 51](#)). The offset (in bytes) of the region's first address is specified in the MKey context.

[Figure 51](#) illustrates an example of a zero-based memory region of 8Kbyte starting at offset of 0.5Kbyte from the beginning of first aligned page.

Figure 51: Zero Based Region/Window



9.3 Shared Memory Regions

Memory regions that share the same memory translation table (MTT) entries are called shared memory regions. It is up to the software to keep referenced MTT entries allocated as long as there are memory regions referencing it.

9.4 Response to Memory Access Errors

HCA hardware validates memory access rights. If access rights are violated, an InfiniBand compliant error is reported and the QP transitions to the appropriate error state. Access violation errors are synchronous errors for which hardware reports a precise exception.

Aside from address translation and protection access errors, transactions may terminate unsuccessfully due to system bus errors. If a posted write incurs one of these latter types of error, it is *not* guaranteed that a subsequent posted write is *not* partially or fully executed.

However, if a bus error occurs during a descriptor read or a data read, it *is* guaranteed that no corrupt data is sent to the network or written to local memory; execution is halted, and the QP is transferred to the appropriate error state.

9.5 Reserved LKey

The HCA supports the use of a reserved LKey. When accessing memory using the reserved LKey, one-to-one mapping between virtual and physical addresses is done. The use of this key is qualified by the *rlky* bit in the QP/RQ/SQ contexts. HCA HW supports multiple Reserved LKeys, one for each virtual HCA. The LKey is programmed by the **Level 0** privilege agent (for example VMM) at device initialization. The value of the reserved LKey is retrieved from the channel adapter by the QUERY_SPECIAL_CONTEXTS command executed by a **Level 1** privilege agent.

9.6 Address Translation Control Structures

9.6.1 MKey Context

The MKey context is configured by the host software executing CREATE_MKEY command (see “CREATE_MKEY – Create MKey Entry” on page 1861). Table 362, “MKey Context Format” shows format of the MKey context argument of this command while operating in enhanced mode.

Table 362 - MKey Context Format

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																							
	sf		ma_translation_mode		tunneled_atomic		access_mode_1_0		lr		lw		rr		arw		small_fence_on_rdma_read_response		relaxed_ordering_write								access_mode_4_2		free																										
qpn																															mkey[7:0]																								
pasid																																																							

Table 362 - MKey Context Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
length64		bsf_en				expected_sigerr_count		en_rinval				pd																		0Ch					
start_addr																																10-14h			
len																																18-1Ch			
bsf_octword_size																																20h			
																BSF handle [47:32]																24h			
BSF_handle[31:0]																																28h			
																mtt handle [47:32]																2Ch			
mtt_handle[31:0]																																30h			
translations_octword_size																																34h			
																										relaxed_ordering_read				log_entity_size				38h	
				crypt-to_en																Generation Counter										3Ch					

The fields of the MKey context is shown in [Table 363](#).

Table 363 - MKey Context Fields

Offset	Bits	Name	Description	Access
00h	30	free	0 - memory key is in use. It can be used for address translation 1 - memory key is free. Cannot be used for address translation	
	28:26	access_mode_4_2	3 MSB bits of access mode. See encoding in access_mode_1_0 field.	
	18	relaxed_ordering_write	If set, issue PCIe cycle with "relaxed ordering" attribute (allows write after write bypassing) for writes. Supported only when HCA_CAP .relaxed_ordering_write==1.	
	16	small_fence_on_read_response	Responder will have one read request toward the PCIe (per QP). bit[1]- goto_pci - If cleared, no PCI access will be performed on the target address bit[2] - relaxed_ordering#. If cleared, PCIe cycle should be issued with relaxed ordering attribute. bit[3] - no_snoop#. If cleared, issue PCIe cycle with "no Snoop" attribute and this cycle will not be snooped in CPU cache bit[4-5] - TPH - TLP processing hints bit[6] - TLP_Processing_hints_en bit[7] - reserved	
	15	umr_en	Enable umr operation on this MKey	
	14	a	If set, atomic operation is enabled	
	13	rw	If set, remote write is enabled	
	12	rr	If set, remote read is enabled	
	11	lw	If set, local write is enabled	
	10	lr	If set, local read is enabled. Must be set for all MKeys	
	9:8	access_mode_1_0	2 LSB bits of access mode. 0x0: PA - (VA=PA, no translation needed) if set, no virtual to physical address translation is performed for this MKey. Not valid for, block mode MKey, replicated MTT MKey 0x1: MTT - (PA is needed) 0x2: KLMs - (Indirect access) 0x3: Fixed_Buffer_Size - used when KLMs are associated with fixed memory size. The fixed sized is determined by log_entity_size. Supported only if .fixed_buffer_size==1. See Table 1571 , "HCA Capabilities Layout," on page 1671 0x4: SW_ICM -Supported only when SW_ICM in HCA_CAP .general_obj_types is set. 0x5: MEMIC - Supported only when HCA_CAP .DEVICE_MEMORY.memic==1. See Table 1605 , "Device Memory Capabilities Layout," on page 1785	

Table 363 - MKey Context Fields

Offset	Bits	Name	Description	Access
00h	6	tunneled_atomic	If set, part of the MSB's of the address is set to (TBD- close with IBM) in order to generate a command in the CPU and do atomic for RDMA WR operations in the memory controller instead of accessing the user's memory (DMA). This is according to IBM p9 spec. Supported only when HCA_CAP.tunneled_atomic==1.	
	5:4	ma_translation_mode	MA translation mode 0x0: None - no MA translation needed. 0x1: ATS - Supported only when HCA_CAP.ats==1, and valid only when access_mode==PA or MTT. 0x3: CAPI - valid when access_mode==PA. Then a CAPI translation can be performed.	
04h	31:8	qpn	For type 2 memory window - indicates the QP number this memory window is attached to. otherwise - must be 0xffffffff.	
	7:0	mkey[7:0]	Variant part of MKey specified by this MKey context	
08h	23:0	pasid	PASID associated with MKEY. Valid when ma_translation_mode==CAPI	
0Ch	31	length64	Enable registering 2 ⁶⁴ bytes per region	
	30	bsf_en	Enable having bsf list on this MKey	
	26	expected_sigerr_count	LSB of expected number of sigerr_cqes. (See Section 17.4.1, "Signature Handover Steps," on page 898)	
	24	en_rinval	Enable remote invalidation	
	23:0	pd	Protection domain	
10h-14h	64	start_addr	Start Address - Virtual address where this region/window starts. For access_mode==MEMIC, indicates the address of an allocated MEMIC buffer.	
18h-1Ch	64	len	Region length. Reserved when length64 bit is set (in which case the region length is 2 ⁶⁴ B). For access_mode==MEMIC, indicates the size of an allocated MEMIC buffer.	
20h	31:0	bsf_octword_size	Size (in units of 16B) required for this MKey's BSFs Must be a multiple of 4	

Table 363 - MKey Context Fields

Offset	Bits	Name	Description	Access
34h	31:0	translations_octword_size	Size (in units of 16B) required for this MKey's physical buffer list or SGEs access_mode: MTT - each translation is 8B access_mode: KLM/Fixed Buffer Size - each SGE is 16B access_mode: PA - reserved Must be a multiple of 4	
38h	6	relaxed_ordering_read	<u>If set, issue PCIe cycle with "relaxed ordering" attribute for reads.</u> Supported only when <u>HCA_CAP.relaxed_ordering_read</u> ==1.	
	4:0	log_entropy_size	When access_mode==MTT: log2 of Page size in bytes granularity. When access_mode==Fixed_Buffer_Size: determines the length of the memory buffers (KLMs) associated with the MKEY. otherwise: reserved. Must be >=12	
	64	bsf_handle	pointer to the location of the bsfs (in ICM)	
	64	mtt_handle	pointer to the location of the mttts (in ICM)	
3C	28:27	crypto_en	0x0: CRYPTO_DISABLED 0x1: CRYPTO_ENABLED	

The following table describes the usage of en_rinval bit and umr bit in the MKC (en_rinval must be cleared for shared Memory Regions).

Table 364 - Umr_en and en_rinval Bits Usage

Region/Window Type	Fast Registration	Local/Remote Invalidate
Memory Regions created through Register MR, Reregister MR (CI owned Key)	Not allowed - MKey.umar_en should be cleared	Not allowed - en_rinval bit should be cleared
Physical Memory Regions created through Register Physical MR, Reregister Physical MR (CI owned Key)	Not allowed - MKey.umar_en should be cleared	Allowed if region is not shared - en_rinval bit should be set
Physical Memory Regions created through Register Physical MR, Reregister Physical MR (Consumer owned Key)	allowed - umr_en bit should be set	Allowed if region is not shared - MKey.en_rinval should be set

The MTT Entry contains the physical tag. An entry of the Memory Translation is shown in [Table 365](#).

Table 365 - Memory Translation Table (MTT) Entry Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ptag[63:32]																															00h	
ptag[31:8]																											wr_en	rd_en	04h			

Table 366 - Memory Translation Table (MTT) Entry

Offset	Bits	Name	Description	Access
00h	31:0	ptag[63:32]	High-order bits of physical tag. The size of the field depends on the page size of the region.	
04h	31:8	ptag[31:8]		
	1	wr_en	Write enable. Writing to pages with wr_en=0 triggers page faults for ODP-supported operations. Must be set to “1” for non-ODP supported operations	
	0	rd_en	Read enable. Reading to pages with rd_en=0 will trigger page faults for ODP-supported operations. Must be set to “1” for non-ODP supported operations	

9.7 Memory Key Configuration (Creation)

Memory Key object is initialized (created) by the privileged SW entity using the CREATE_M-KEY command.

10 Firmware Components Update

10.1 Overview

The device supports a safe method for updating its firmware components using tools or the device driver. The update procedure allows:

- Maintaining access exclusiveness to a component
- Flash storage area erasure
- Correct placement of components data in the flash storage
- Verification and activation of the components
- Query of existing components

Note: Legacy tools collect the device and running firmware information directly from the flash by providing explicit addresses for reading to MFBA command (according to the flash layout listed in [FLASH_PROG]). Conversely, the MCQI and MHWI does not require the caller to provide flash device addresses.

10.2 Firmware Components

The device has the following firmware components:

1. Boot image - the device firmware. It may also include a PCI expansion ROM with Flexboot (for servers running BIOS) and/or UEFI drivers and applications.

The supported FW components and their status are discovered by querying “[MCQS - Management Component Query Status](#)” register. The *component_index* should be scanned sequentially, starting at index 0 and ending when *last_index_flag* is set.

10.2.1 Component Update State

A component may be in one of the following update states:

- IDLE - component storage is not erased.
- IN_PROGRESS - component is being updated.
- APPLIED - component is ready for activation.
- ACTIVE - component is in use.
- ACTIVE_PENDING_RESET - component will be in use following device reset.
- FAILED - last update attempt has failed.
- CANCELED - last update attempt was canceled by the initiator.
- BUSY - component cannot currently be updated.

The component state is not preserved through boot cycles. On boot, each component is initialized to one of the states: ACTIVE, BUSY, IDLE.

10.2.2 Component Properties

The properties of each component may be queried using “[MCQI - Management Component Query Information](#)” register, referencing a component by its *component_index*. The component has one or more sets of properties, referred to as *info_type*. Each properties set may be read in multiple accesses, each indicating an *offset* inside the property set structure and the amount of requested *data_size*. The actual returned data is indicated by *info_size*.

Note: The requested data may exceed the property set structure size. The response will indicate the amount of valid data returned.

The first set “CAPABILITIES” indicates which other sets are supported for the component in *supported_info_mask* field. In addition, it reports the attributes a component update must have to be applicable for the device in *match_chip_id*, *match_psid* and *match_base_guid_mac* fields. If multiple component updates are available, the correct one should be chosen according to these parameters. The matching device attributes can be queried using the MGIR register.

10.3 Update State Machine

The access to firmware components is controlled by a state machine. The FSM includes a mechanism to the caller to gain exclusive access, see “[Update Handle](#)”. Once exclusivity is guaranteed, the FSM can execute one of three flows:

- “[Update Flow](#)” - Update/load a component
- “[Activation Flow](#)” - Activate one/all updated component(s)
- “[Read Flow](#)” - Read the data of a component

The FSM is managed by the “[MCC - Management Component Control](#)” register. Unless otherwise specified, instructions for handling the FSM refer to this register. For detailed description of the FSM and its states, see “[FSM States](#)” on page 639

10.3.1 Update Handle

The update handle is a token, registered when the FSM is activated, and used for every following action as an identifier of the operation. The update handle is verified by the device, to make sure that operations do not unexpectedly collide. Attempted actions with an invalid/expired handle will fail.

The update handle should be a unique identifier. An initiator of an operation should acquire a unique handle by reading “[MCC - Management Component Control](#)” register, *update_handle* field. This value should be registered back to the device by the LOCK_UPDATE_HANDLE instruction. Once locked, this handle should be repeated in every other instruction to the FSM, or when accessing the component itself, using “[MCDA - Management Component Data Access](#)” register.

When the operation is done, the update handle should be released using the RELEASE_UPDATE_HANDLE instruction. The handle could also be forcefully released by FORCE_HANDLE_RELEASE instruction, when the handle is unexpectedly invalid.

The state of the update handle can be queried using CHECK_UPDATE_HANDLE instruction, to confirm its validity. In case the update handle is no longer valid, it may attempt to lock a new handle and restart the operation.

10.3.2 Update Flow

This flow updates a firmware component. To initiate the flow, an UPDATE_COMPONENT instruction should be issued. The instruction should include the *component_index* identifying the component to be updated. This instruction will start initializing space for the component. Indicating the *component_size* may reduce initialization time.

The FSM will be in “INITIALIZE State” until initialization is over. Then, the FSM will transition to “DOWNLOAD State”. At that point, the new component may be written using “MCDA - Management Component Data Access” register.

Once the new component is completely downloaded to the device, the component should be verified by issuing VERIFY_COMPONENT instruction. The FSM will transition to “VERIFY State” until verification is done. After the component is successfully verified, it will be applied and the FSM will transition to “APPLY State”.

Once the component is applied the update flow is complete and the FSM returns to “LOCKED State”.

10.3.3 Activation Flow

This flow activates the updated components. To initiate the flow for all components that are in APPLIED state, an ACTIVATE instruction should be used. To initiate the flow for a single component an ACTIVATE_COMPONENT instruction should be used, and the *component_index* identifies the component to be activated. The FSM will be in “ACTIVATE State” until activation is over, then it will return to “LOCKED State”.

Different components may require additional actions for activation:

- Automatic activation - Activation flow is not required at all. The component is active once “Update Flow” is complete.
- AC power cycle - The component will become active only after a power cycle for both the PCI power rails and the AUX power.
- DC power cycle - The component will become active only after a power cycle of the PCI power rails.
- Server reboot - The component will become active only after a server reboot.
- Firmware reset - The component will become active only after a firmware reset.
- Multi-Host synchronization - The component will become active only after all hosts perform the required action (of the ones mentioned above) simultaneously.

The required actions for each component may be queried using “MCQI - Management Component Query Information” register, under ACTIVATION_METHOD properties set.

10.3.4 Read Flow

This flow allows reading a component. To initiate the flow for reading a component a `READ_COMPONENT` instruction should be used. for reading the pending version of a component a `READ_PENDING_COMPONENT` instruction should be used. The FSM will transition to “[UPLOAD/UPLOAD_PENDING State](#)” according to the instruction. At that point, the component may be read using “[MCDA - Management Component Data Access](#)” register.

Some components may not be read. This property of the component is indicated by the *rd_en* field in “[MCQI - Management Component Query Information](#)” register.

The flow is terminated by issuing the `CANCEL` instruction.

10.3.5 FSM Monitoring

The FSM state can be queried by reading the “[MCC - Management Component Control](#)” register *control_state* field. For some states, where a procedure is performed by the device, the progress is reported by the *control_progress* field. In case the procedure fails, and the FSM cannot proceed, it will be reported by *error_code* field. See “[Error Handling](#)” on page 638 for more details.

10.3.6 Flow Cancellation

At any point in time, a flow already started may be terminated. The `CANCEL` instruction stops the flow and returns the FSM to “[LOCKED State](#)”. The update handle remains valid, and a new flow may be initiated.

In addition, `RELEASE_UPDATE_HANDLE` and `FORCE_HANDLE_RELEASE` instructions also terminate an active flow, returning the FSM to “[IDLE State](#)” and releasing the update handle.

10.3.7 Error Handling

Upon Error, the FSM remains in the problematic state. The FSM can be released by:

- Canceling the current flow (See “[Flow Cancellation](#)” on page 638).
- Releasing the update handle completely, using `FORCE_HANDLE_RELEASE` instruction.

The error cause is reported in *error_code* field. The detected errors may be one of the following:

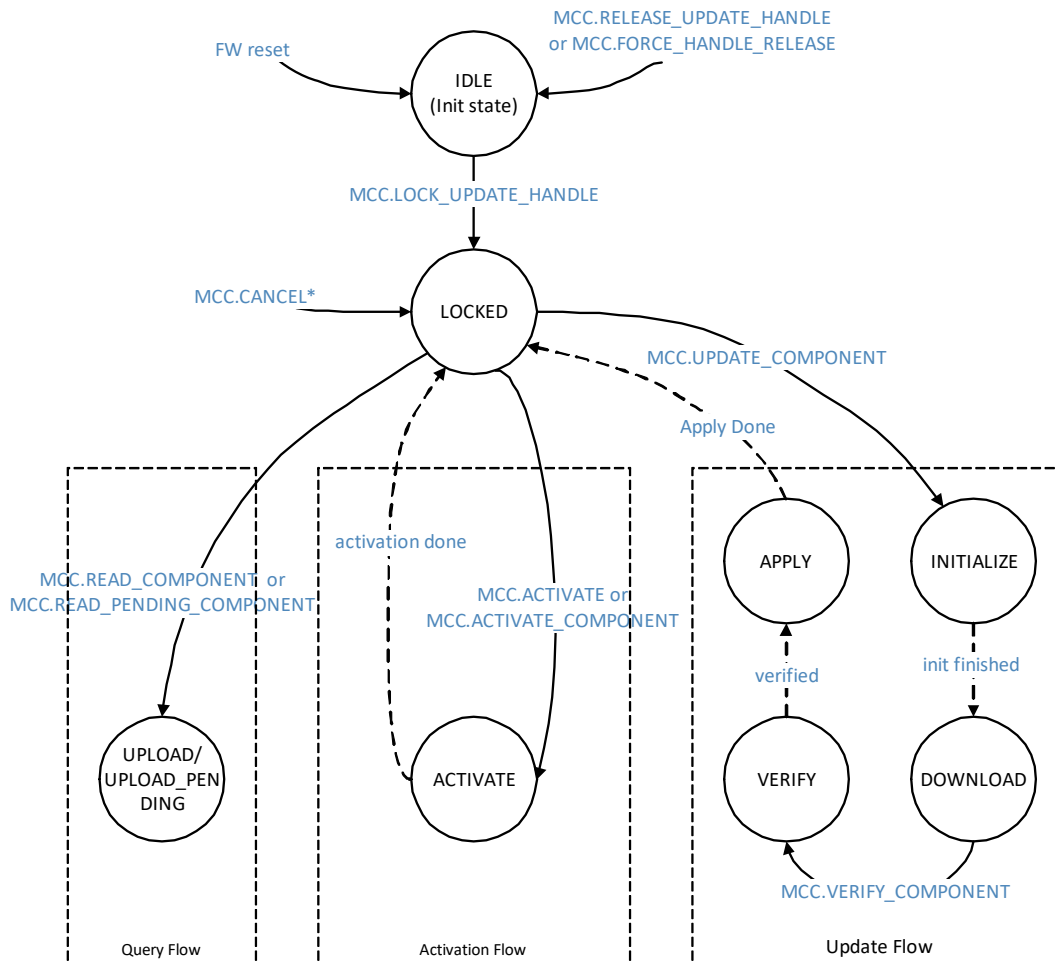
- `ERROR` -- an unspecified error
- `REJECTED_DIGEST_ERR` - Verification of the newly downloaded component has failed because the calculate hash value does not match the hash value included in the component.
- `REJECTED_NOT_APPLICABLE` - Verification of the newly downloaded component has failed because the component is not applicable for this device.
- `REJECTED_UNKNOWN_KEY` - Verification of the newly downloaded component has failed because the running firmware does not have the public-key needed for validating the component signature.

- REJECTED_AUTH_FAILED - Verification of the newly downloaded component has failed because the component has failed to pass signature validation.
- REJECTED_UNSIGNED - Verification of the newly downloaded component has failed as the component has a hash value instead of a proper signature.

10.3.8 FSM States

Figure 52 describes the state diagram of the FW component update FSM. The dashed state transitions are done by the device when the procedure performed in the current state completes successfully. The states are detailed below.

Figure 52: FW Component FSM State Diagram



10.3.8.1 IDLE State

In the IDLE state, no operation is in progress and the update handle is unlocked.

This is the initial state after power up, and the state entered after when the update handle is released, either by `RELEASE_UPDATE_HANDLE` instruction, or forcefully by the `FORCE_HANDLE_RELEASE` instruction.

On entry to IDLE state, all components that are not in a stable state (BUSY, ACTIVE or ACTIVE_PENDING_RESET) to IDLE, to clear any incomplete operations.

In this state, an operation may be initiated by registering an update handle using the `LOCK_UPDATE_HANDLE` instruction, and transitioning to “[LOCKED State](#)”.

10.3.8.2 LOCKED State

This state is entered after a successful `LOCK_UPDATE_HANDLE` instruction from “[IDLE State](#)”, or after canceling an on-going flow using `CANCEL` instruction.

In this state, any of the flows may be initiated:

- “[Update Flow](#)” by issuing `UPDATE_COMPONENT` instruction and transitioning to “[INITIALIZE State](#)”
- “[Activation Flow](#)” - by issuing `ACTIVATE` or `ACTIVATE_COMPONENT` instructions and transitioning to “[ACTIVATE State](#)”
- “[Read Flow](#)” - by issuing `READ_COMPONENT` instruction and transitioning to “[UPLOAD/UPLOAD_PENDING State](#)”

10.3.8.3 INITIALIZE State

This state is entered when the handle owner selects a specific component for updating, by sending an `UPDATE_COMPONENT` instruction from “[LOCKED State](#)”.

In this state, the required space for the component, specified in *component_size*, is initialized. As this can be a long process, the *control_progress* field can be queried to report progress.

When initialization is done, the component state is modified to INITIALIZED and the FSM transitions to “[DOWNLOAD State](#)”.

The `UPDATE_COMPONENT` instruction may fail if the component is in BUSY state.

10.3.8.4 DOWNLOAD State

This state is entered when component initialization is done. In this state, the component may be updated by writing to “[MCDA - Management Component Data Access](#)” register.

When update is complete, the component can be verified using `VERIFY_COMPONENT` instruction, and transitioning to “[VERIFY State](#)”.

10.3.8.5 VERIFY State

This state is entered when the handle owner initiates `VERIFY_COMPONENT` instruction from “[DOWNLOAD State](#)”. In this state, the integrity of the downloaded component is checked. On success, the FSM immediately transitions to “[APPLY State](#)”.

10.3.8.6 APPLY State

This state is entered immediately after a successful verification of the component in “[VERIFY State](#)”. In this state the updated component is applied. On successful application, the component state is changed to APPLIED, and the FSM transitions to “[LOCKED State](#)”.

10.3.8.7 ACTIVATE State

This state is entered when the handle owner issues ACTIVATE or ACTIVATE_COMPONENT instructions from “[LOCKED State](#)”. In this state the components in APPLIED state are activated. If ACTIVATE instruction was used, all components that are in APPLIED state are activated. If ACTIVATE_COMPONENT was used, only the specified component in *component_index* will be activated.

10.3.8.8 UPLOAD/UPLOAD_PENDING State

This state is entered when the handle owner issues READ_COMPONENT or READ_PENDING_COMPONENT instructions. In this state a component may be read by reading the “[MCDA - Management Component Data Access](#)” register.

10.4 Configuration notes

Firmware components may be updated through multiple interfaces. Through each interface, the support for this feature is indicated differently:

- Command Interface - support is indicated by *fw_ctrl_update* field in QUERY_HCA_CAP
- ICMD Interface - support is indicated by *fw_ctrl_update* field in ICMD_QUERY_CAPABILITY
- MAD Interface - attempts to access the relevant registers will fail.

11 Non-Volatile Configuration (NVConfig)

The device has a designated area in an on-board flash to provide storage for non-volatile configuration options. Enabling configuration options such as Wake-on-LAN and SR-IOV can be set in persistent storage, and affect the device operation.

Each NV configuration may target one of the following entities:

- The device (NIC) - configurations applied by the device at the boot sequence, following a device reset (see [Section 28.32.2, “MFRL - Management Firmware Reset Level,”](#) on [page 1645](#) for information regarding device reset)
- The Expansion ROM (EXP_ROM) - configurations applied by the Expansion ROM that runs on the server during per-boot stage (Flexboot/PXE for Legacy BIOS or UEFI)
- **[Internal]** Internal device configuration (NIC_internal) - Same as NIC configuration. However, the configurations are updated internally by the device.

11.1 Configuration Interface

In order to access the non-volatile configuration area, a driver should use “[Access Registers](#)” command. The MNVGC command is used to query global capabilities of the NV infrastructure (that is, capabilities that are not unique for a specific type).

Specific registers allow reading, writing, and invalidating configuration items. “[MNVDA - Non-Volatile Data Access](#)” register can be used to read and write configuration items. “[MNVDI - Non-Volatile Data Invalidate](#)” register can be used to invalidate (delete) a specific configuration item, and “[MNVIA - Non-volatile data Invalidate All](#)” register can be used to invalidate all configuration items and revert the device to its factory default settings.

[Internal] In some cases, a configuration write will require modification of multiple items atomically, so they either all have effect or none of them do. TBD This can be done by writing the items with the *shadow* field set in the NVDA header. The write only takes effect later when “[MNVCT - Non-volatile commit](#)” register is used to commit the series of writes.

The HCA exposes the supported set of configuration items as part of the output of “[MNVQC - Non-volatile query capabilities](#)” register.

Note: MNVQC is supported for configuration consumed by the NIC itself. MNVQC for configuration consumed by the NIC's PCI expansion ROM (UEFI drivers Flexboot/PXE boot driver) are not available via the MNVQC commands. Alternatively, the configuration capabilities of the expansion ROM are expressed by the NV_ROM_INI and NV_ROM_CAP TLVs that are read using the MNVDA commands.

In order to allow tools to work concurrently with the driver, non-volatile configuration access is also implemented through the ICMD interface using “[ICMD_ACCESS_REGISTER - Register Access](#)”. The support for this interface is indicated by *nv_access* field in “[ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities](#)”.

11.2 Defaults

The device includes defaults for most non-volatile configuration parameters. **OEMs can modify the default by changing the configuration file (.INI) and rebuild the firmware (TBD: should we add a description of the process).**

The defaults of a given TLV can be read by the MNVDA read command with the 'factory settings' flag set. The application must first use the MNVGC command to detect if this capability is supported.

11.3 NV Configurations Privileges

Many of the NV configuration options affect the entire device or the physical port. In Multi-Host environments, it may be useful to isolate between the hosts, and leave the global configurations in the hands of one of them. This may be achieved by configuring NV_HOST_PRIV_CONF from one host, and reducing the privileges of the other hosts.

A host with reduced privileges may not modify Global and Port configurations. In addition, it may not modify the relevant per-host configurations, if a privileged host has previously written them. As a result, a reduced capabilities host may not use MNVIA as well.

The host privilege level takes effect immediately, unlike other configurations, which only take effect after reboot.

11.4 Lifecycle and Defaults [Unimplemented]

The device exposes different configuration settings at the different stages of the product's lifecycle. The device comes out of the factory set to the Mellanox lifecycle stage. At this stage, all of the device's configuration may be set. The device is then set to the OEM stage and shipped to OEMs. OEMs may customize some of the settings, either by changing the default values or by prohibiting the user from overriding some settings altogether. The OEM sets the stage to the end-user stage and ships it to the end user. The end-user may further customize the device within the bounds set by the previous stages.

Table 367 - Lifecycle Summary

Priority	code	Description
Mellanox	2	Intended to be written during manufacturing or during firmware upgrade. May contain Mellanox controlled default values and settings.
OEM	1	Intended to be written during OEM manufacturing stage but locked from end-user changes.
End-User	0	Settings that can be controlled by the end-user.

Each configuration setting is written to the flash along with the code of the lifecycle stage during which it was written. In addition, the configuration setting determines which other stages will be able to further modify this setting. The *ovr_en* field determines whether stages with a lower code may override the configuration. The *rd_protect* field determines whether stages with a lower code number may read this setting.

For example, if *ovr_en* field is set for a configuration setting during *Mellanox* stage, OEMs will be able to override that setting. An end-user would be able to do so too, unless the OEM has cleared the *ovr_en* field.

11.5 Conventions

11.5.1 Classes

Table 368 - NVConfig Classes

Class	Class Code	Description
Global	0	Global parameters (shared between all hosts, functions and ports)
Physical Port	1	Parameters specific to one of the devices' physical ports
BMC	2	Settings that are specific to one of the BMC's that are connected to the device (a multi-host system may have one per-host and another chassis BMC).
Per-host/Function	3	Settings which are specific to a single host and a single physical function on that host.
File	6	
Host	7	Settings which are specific to a single host

11.5.2 Naming Guidelines for Mlxconfig

1. Addressing per port parameter is possible using the "_P1" and "_P2" suffixes. the parameter names without the suffix will configure the port associated with the function.
2. A name of a parameter that only has one field is the parameter name with the 'nv_tlv_' prefix.
3. The 'CONFIG_FOR_PF=N' directive tells mlxconfig to start using the specified physical function for configuring the rest of the parameters in a configuration file. This is mainly applicable for class=3 configuration. This parameter will be expressed in a meta-data TLV that mlxconfig will not try to burn.

11.5.3 Tags, Attributes and Primitives

The following frame-maker's tags will be used:

1. "class" tag on the 3 summary tables will indicate the class of the listed TLVs (from Table 368, "NVConfig Classes," on page 644)

2. “adabe-name” tag on the per-parameter tables will include the ADABE name for the parameter.
3. mlxconfig' tag in the description column of the per-parameter table will mark multiple rows that will describe the mlxconfig ADABE attributes listed on "Table 369, "Attributes for the Generic Mlxconfig Marked with the 'Mlxconfig' Tag". The Ayme script will extract the attributes from the 'mlxconfig' tagged text. Each attributes starts in a new line - so search is easy.

Table 369 - Attributes for the Generic Mlxconfig Marked with the 'Mlxconfig' Tag

Attribute	Description	
mlxconfig_desc	Help for the parameter to be displayed by mlxconfig. If none defined, mlxconfig will use the PRM description	Internal PRM
mlxconfig_name:	The name of the field in mlxconfig. a parameter without a name is not configurable by mlxconfig	Internal PRM
xuefi	Name of the parameter in HII/ X-UEFI protocol	Internal PRM
supported_from_version:	The initial TLV version this parameter was supported from. If not present, the tool assumes that the parameter was introduced in version 0.	PRM

Table 370 - Variable Types

TYPE	Range	Default display format	Notes
BOOL	0..1	TRUE FALSE	One bit field inside a DWORD. Serialization for NVDA and binary configuration files is done in big-endian format.
INTEGER	—(2<field-len>-1) to (2 <field-len> - 1)	Decimal	A field with a size of 1-32 bits inside a DWORD. Serialization for NVDA and binary configuration files is done in big-endian format.
UNSIGNED	0-(2<field-len>-1)	Decimal	A field with a size of 1-32 bits inside a DWORD. Serialization for NVDA and binary configuration files is done in big-endian format.
BINARY	0-(2<field-len>-1)	Hexadecimal (without leading zeros)	A field with a size of 1-32 bits inside a DWORD. Serialization for NVDA and binary configuration files is done in big-endian format.

Table 370 - Variable Types

TYPE	Range	Default display format	Notes
ENUM		ENUM names	ENUM values are listed in the description column in the following format: 0x<hex-value>: <enum-name> [- <enum-description>] where <enum-name> must start with an alpha character. If the enum name should start with a digit, a leading underscore can be used. (which will be ignored by mlxconfig UI). Serialization for NVDA and binary configuration files is done in big-endian format.
STRING	Bounded or unbounded.	NULL terminated string (or full length)	A zero terminated string. The field length must be DWORD aligned (4,8,12,16..). If this is the last parameter in an NVCONFIG TLV the length can be unbounded and the actual length will be derived from the NULL location or from the TLV length.
BYTES	field length	Hexadecimal	The field length must be DWORD aligned (4,8,12,16..). If this is the last parameter in an NVCONFIG TLV, the length can be unbounded and the actual length will be derived from the TLV length.

11.6 NVConfig Summary

11.6.1 Global Parameters

Table 371 - Global Parameters [class_type=0]

Parameter Name	Index	Target	Cap	Version	Reference
NV_BASE_MAC_GUID	0x02	NIC		0	See Table 376, “NV_BASE_MAC_GUID Layout,” on page 657
NV_BASE_MAC_GUID_CAP	0x03	NIC	V	0	See Table 378, “NV_BASE_MAC_GUID_CAP Layout,” on page 658
NV_FORBIDDEN_VERSION	0x04	NIC		0	See Table 380, “NV_FORBIDDEN_VERSIONS Layout,” on page 658
NV_MEMIC_CONF	0x06	NIC		0	See Table 382, “NV_MEMIC_CONF Layout,” on page 660
NV_MEMIC_CAP	0x07	NIC	V	0	See Table 384, “NV_MEMIC_CAP Layout,” on page 660

Table 371 - Global Parameters [class_type=0]

Parameter Name	Index	Target	Cap	Ver sion	Reference
NV_HOST_CHA INING_CONF	0x08	NIC		0	See Table 386, “NV_HOST_CHAINING_CONF Lay- out,” on page 661
NV_HOST_CHA INING_CAP	0x09	NIC	V	0	See Table 388, “NV_HOST_CHAINING_CAP Layout,” on page 662
NV_FPGA_- CONF	0x0A	NIC		0	See Table 390, “NV_FPGA_CONF Layout,” on page 663
NV_FPGA_CAP	0x0B	NIC	V	0	See Table 392, “NV_FPGA_CAP Layout,” on page 664
NV_INTER- NAL_CPU_- CONF	0x0C	NIC		0	See Table 394, “NV_INTERNAL_CPU_CONF Layout,” on page 664
NV_INTER- NAL_CPU_CAP	0x0D	NIC	V	0	See Table 396, “NV_INTERNAL_CPU_CAP Layout,” on page 665
NV_FLEX- _PARS_CONF	0x0E	NIC		0	See Table 398, “NV_FLEX_PARSER_CONF Layout,” on page 665
NV_FLEX- _PARS_CAP	0x0F	NIC	V	0	See Table 400, “NV_FLEX_PARSER_CAP Layout,” on page 666
NV_ROCE_1_5_- CONF	0x10	NIC		0	See Table 516, “NV_ROCE_1_5_CONF Layout,” on page 736
NV_HOST_TO_ BMC	0x11	NIC		0	See Table 518, “NV_HOST_TO_BMC Layout,” on page 737
NV_SAFE_- MODE_PETEK	0x12				TBD
NV_INTER- NAL_HAIRPIN_- CONF	0x13	NIC		0	See Table 520, “NV_INTERNAL_HAIRPIN_CONF Layout,” on page 737
NV_INTER- NAL_HAIRPIN_- CAP	0x14	NIC	V	0	See Table 522, “NV_INTERNAL_HAIRPIN_CAP Lay- out,” on page 738
NV_GLOBAL_P- CI_CONF	0x80	NIC		0	See Table 430, “NV_GLOBAL_PCI_CONF Layout,” on page 682
NV_GLOBAL_P- CI_CAP	0x81	NIC	V	0	See Table 432, “NV_GLOBAL_PCI_CAP Layout,” on page 686
NV_TPT_CONF	0x82	NIC		0	See Table 442, “NV_TPT_CONF Layout,” on page 701
NV_GLOBAL_P- CI_CONF_2	0x84	NIC		0	See Table 436, “NV_GLOBAL_PCI_CONF_2 Layout,” on page 694

Table 371 - Global Parameters [class_type=0]

Parameter Name	Index	Target	Cap	Version	Reference
NV_KDNET_DATA	0x85	NIC		0	See Table 506, “NV_KDNET_DATA Layout,” on page 732
NV_PCI_SWITCH_EVB_CONF	0x86	NIC		0	See Table 508, “NV_PCI_SWITCH_EVB_CONF Layout,” on page 733
NV_PCI_SWITCH_EVB_CAP	0x87	NIC	V	0	See Table 510, “NV_PCI_SWITCH_EVB_CAP Layout,” on page 734
NV_POWER_CONF	0x88	NIC		0	See Table 562, “NV_POWER_CONF Layout,” on page 763
NV_POWER_CAP	0x89	NIC	V	0	See Table 564, “NV_POWER_CAP Layout,” on page 765
NV_HOST_SYNC_CONF	0x8A	NIC		0	See Table 566, “NV_HOST_SYNC_CONF Layout,” on page 766
NV_EMULATION_NVME_CONF	0x8B	NIC		0	See Table 568, “NV_EMULATION_NVME_CONF Layout,” on page 767
NV_EMULATION_NVME_CAP	0x8C	NIC	V	0	See Table 570, “NV_EMULATION_NVME_CAP Layout,” on page 768
NV_EMULATION_PCI_SWITCH_CONF	0x8D	NIC		0	See Table 572, “NV_EMULATION_PCI_SWITCH_CONF Layout,” on page 769
NV_EMULATION_PCI_SWITCH_CAP	0x8E	NIC	V	0	See Table 574, “NV_EMULATION_PCI_SWITCH_CAP Layout,” on page 770
NV_EMULATION_VIRTIO_NET_CONF	0x90	NIC		0	See Table 576, “NV_EMULATION_VIRTIO_NET_CONF Layout,” on page 770
NV_EMULATION_VIRTIO_NET_CAP	0x91	NIC	V	0	See Table 578, “NV_EMULATION_VIRTIO_NET_CAP Layout,” on page 772
NV_EMULATION_VIRTIO_BLK_CONF	0x92	NIC		0	See Table 580, “NV_EMULATION_VIRTIO_BLK_CONF Layout,” on page 773

Table 371 - Global Parameters [class_type=0]

Parameter Name	Index	Target	Cap	Version	Reference
NV_EMULATION_VIRTIO_BLK_CAP	0x93	NIC	V	0	See Table 582, “NV_EMULATION_VIRTIO_BLK_CAP Layout,” on page 775
NV_GLOBAL_PCI_CONF_3	0x94	NIC		0	See Table 438, “NV_GLOBAL_PCI_CONF_3 Layout,” on page 695
NV_DEBUG_MODE	0xFF	NIC		0	See Table 470, “NV_DEBUG_MODE Layout,” on page 720
NV_ROM_INI	0x100	EXP_ROM	V	0	See Table 588, “NV_ROM_INI Layout,” on page 778
NV_ROM_CAP	0x101	EXP_ROM	V	0	See Table 590, “NV_ROM_CAP Layout,” on page 782
NV_GLOBAL_HM_RESTORE_PARAMS	0x107	NIC_INTERNAL			OCBB, OCSD, rate limit info
NV_SW_OFFLOAD_CONFIG	0x10A	NIC		1	See Table 444, “NV_SW_OFFLOAD_CONFIG Layout,” on page 702
NV_SW_OFFLOAD_CAP	0x10B	NIC	V	0	See Table 446, “NV_SW_OFFLOAD_CAP Layout,” on page 707
NV_PACKET_PACING	0x10C	NIC		0	See Table 448, “NV_PACKET_PACING Layout,” on page 710
NV_FILE_ID_MLNX	0x10E	NIC		0	See Table 450, “NV_FILE_ID_MLNX Layout,” on page 710
NV_FILE_ID_VENDOR	0x10F	NIC		0	See Table 452, “NV_FILE_ID_VENDOR Layout,” on page 711
NV_FPGA_DATA	0x110	NIC		0	See Table 512, “NV_FPGA_DATA Layout,” on page 735
NV_PROG_CC	0x111	NIC		0	See Table 454, “NV_PROG_CC Layout,” on page 712
~~NV_HPCC_INT_GLOBAL_PREF_RATE_LIMITER_DEBUG~~	0x112	~~NIC-NIC_INTERNAL~~		0	~~See Table 456, “NV_HPCC_INT Layout,” on page 713~~
NV_GLOBAL_PCI_CAP_3	0x113	NIC	V	0	See Table 434, “NV_GLOBAL_PCI_CAP_3 Layout,” on page 690
NV_HPCC_INT	0x112 ~~4~~	NIC		0	See Table 456, “NV_HPCC_INT Layout,” on page 713
<u>NV_HPCC_INT_CAP</u>	<u>0x115</u>	<u>NIC</u>	<u>V</u>	<u>0</u>	See Table 458, “NV_HPCC_INT_CAP Layout,” on page 715

Table 371 - Global Parameters [class_type=0]

Parameter Name	Index	Target	Cap	Version	Reference
NV_IB_DC_CONF	0x190	NIC		1	See Table 460, “NV_IB_DC_CONF Layout,” on page 716
NV_IB_DC_CAP	0x191	NIC	V	0	See Table 462, “NV_IB_DC_CAP Layout,” on page 716
NV_ENCLOSURE_INFO	0x192	NIC		0	See Table 464, “NV_ENCLOSURE_INFO Layout,” on page 717
NV_CURRENT_CONFIGURATOR	0x194	NIC			See Table 466, “NV_CURRENT_CONFIGURATOR Layout,” on page 718
NV_INVALIDATE_ALL_TIMESTAMP	0x19A	NIC		0	See Table 468, “NV_INVALIDATE_ALL_TIMESTAMP Layout,” on page 719
NV_SECURE_HOST	0x19B	NIC		0	See Table 504, “NV_SECURE_HOST Layout,” on page 731
NV_ROM_OCSD_OCBB	0x211	EXP_ROM		0	See Table 610, “NV_ROM_OCSD_OCBB Layout,” on page 800
NV_GLOBAL_FW_INT_0	0x3F0	NIC_INTERNAL		0	Internal for FW use
NV_VTS_BANK_A	0x400	NIC		0	See Table 584, “NV_VTS_BANK_A Layout,” on page 776
NV_VTS_BANK_B	0x401	NIC		0	See Table 586, “NV_VTS_BANK_B Layout,” on page 776

11.6.2 Per Port Parameters

Table 372 - Per-Port Parameters [class_type=1]

Parameter Name	Index	Target	Cap	Version	Reference
NV_VPI_LINK_TYPE	0x12	NIC		0	See Table 428, “NV_VPI_LINK_TYPE Layout,” on page 681
NV_ROCE_CC	0x107	NIC		0	See Table 528, “NV_ROCE_CC Layout,” on page 742
NV_ROCE_C_C_ECN	0x108	NIC		0	See Table 530, “NV_ROCE_CC_ECN Layout,” on page 742
NV_ROCE_C_C_QCN	0x109	NIC		0	See Table 532, “NV_ROCE_CC_QCN Layout,” on page 746

Table 372 - Per-Port Parameters [class_type=1]

Parameter Name	Index	Target	Cap	Ver sion	Reference
NV_LLDP_NB_- CONF	0x10A	NIC		1	See Table 524, “NV_LLDP_NB_CONF Layout,” on page 739
NV_LLDP_NB_- CAP	0x10B	NIC	V	1	See Table 526, “NV_LLDP_NB_CAP Layout,” on page 740
NV_LLDP_N- B_TX_CHAS- SIS_ID_TLV	0x110	NIC		0	See Table 474, “NV_LLDP_NB_TX_CHASSIS_ID_TLV Layout,” on page 721
NV_LLDP_N- B_TX- PORT_ID_TLV	0x111	NIC		0	See Table 476, “NV_LLDP_NB_TX_PORT_ID_TLV Layout,” on page 722
NV_LLDP_N- B_TX_TTL_TLV	0x112	NIC		0	See Table 478, “NV_LLDP_NB_TX_TTL_TLV Layout,” on page 722
NV_LLDP_N- B_TX_OPTION- AL_TLV0	0x113	NIC		0	See Table 480, “NV_LLDP_NB_TX_OPTION- AL_TLV0 Layout,” on page 723
NV_LLDP_N- B_TX_OPTION- AL_TLV1	0x114	NIC		0	See Table 482, “NV_LLDP_NB_TX_OPTION- AL_TLV1 Layout,” on page 723
NV_LLDP_N- B_TX_OPTION- AL_TLV2	0x115	NIC		0	See Table 484, “NV_LLDP_NB_TX_OPTION- AL_TLV2 Layout,” on page 724
NV_LLDP_N- B_TX_OPTION- AL_TLV3	0x116	NIC		0	See Table 486, “NV_LLDP_NB_TX_OPTION- AL_TLV3 Layout,” on page 724
NV_LLDP_N- B_TX_OPTION- AL_TLV4	0x117	NIC		0	See Table 488, “NV_LLDP_NB_TX_OPTION- AL_TLV4 Layout,” on page 725
NV_LLDP_N- B_TX_OPTION- AL_TLV5	0x118	NIC		0	See Table 490, “NV_LLDP_NB_TX_OPTION- AL_TLV5 Layout,” on page 725
NV_LLDP_N- B_TX_OPTION- AL_TLV6	0x119	NIC		0	See Table 492, “NV_LLDP_NB_TX_OPTION- AL_TLV6 Layout,” on page 726
NV_LLDP_N- B_TX_OPTION- AL_TLV7	0x11A	NIC		0	See Table 494, “NV_LLDP_NB_TX_OPTION- AL_TLV7 Layout,” on page 726
NV_LLDP_N- B_DCBX	0x18E	NIC		0	See Table 472, “NV_LLDP_NB_DCBX Layout,” on page 721

Table 372 - Per-Port Parameters [class_type=1]

Parameter Name	Index	Target	Cap	Version	Reference
NV_KEEP_LINK_UP	0x190	NIC		23	See Table 496, “NV_KEEP_LINK_UP Layout,” on page 727
NV_QOS_CONF	0x192	NIC		0	See Table 498, “NV_QOS_CONF Layout,” on page 729
NV_QOS_CAP	0x193	NIC	V	0	See Table 500, “NV_QOS_CAP Layout,” on page 730
NV_MPFS_CONF	0x196	NIC		0	See Table 558, “NV_MPFS_CONF Layout,” on page 760
NV_MPFS_CAP	0x197	NIC	V	0	See Table 560, “NV_MPFS_CAP Layout,” on page 762
<u>NV_PORT_FW_INT_0</u>	<u>0x3F0</u>	<u>NIC INTERNAL</u>		<u>0</u>	<u>Internal for FW use</u>

11.6.3 Per Host-Function Parameters

Table 373 - Per-Host/Function Parameters [class_type=3]

Parameter Name	Index	Target	Cap	Version	Reference
NV_WAKE_ON_LAN_CONF	0x10	NIC		0	See Table 402, “NV_WAKE_ON_LAN_CONF Layout,” on page 668
NV_WAKE_ON_LAN_CAP	0x1F	NIC	V	0	See Table 414, “NV_WAKE_ON_LAN_CAP Layout,” on page 676
NV_VIRT_NET_ADDR	0x110	NIC		0	See Table 416, “NV_VIRT_NET_ADDR Layout,” on page 676
NV_VIRT_GUID_ADDR	0x111	NIC		0	See Table 418, “NV_VIRT_GUID_ADDR Layout,” on page 677
NV_HCA_CONF	0x112	NIC		0	See Table 420, “NV_HCA_CONF Layout,” on page 678
NV_HCA_CAP	0x113	NIC	V	0	See Table 422, “NV_HCA_CAP Layout,” on page 679
NV_ECPF_CONF	0x114	NIC		0	See Table 424, “NV_ECPF_CONF Layout,” on page 679
NV_ECPF_CAP	0x115	NIC	V	0	See Table 426, “NV_ECPF_CAP Layout,” on page 680
NV_EXTERNAL_PORT_CTRL	0x192	NIC		0	See Table 514, “NV_EXTERNAL_PORT_CTRL Layout,” on page 735

Table 373 - Per-Host/Function Parameters [class_type=3]

Parameter Name	Index	Target	Cap	Ver sion	Reference
NV_ROM_- BOOT_CONF2	0x195	EXP_RO M		2	See Table 592, “NV_ROM_BOOT_CONF2 Layout,” on page 784
NV_ROM_UE- FI_CONF	0x196	EXP_RO M		0	See Table 594, “NV_ROM_UEFI_CONF Layout,” on page 784
NV_ROM_DE- BUG_LEVEL	0x202	EXP_RO M		0	See Table 602, “NV_ROM_DEBUG_LEVEL Layout,” on page 791
NV_ROM_- FLEXBOOT_DE- BUG	0x204	EXP_RO M		0	See Table 606, “NV_ROM_FLEXBOOT_DEBUG Lay- out,” on page 799
NV_ROM_UE- FI_DE- BUG_LEVEL	0x206	EXP_RO M		0	See Table 604, “NV_ROM_UEFI_DEBUG_LEVEL Lay- out,” on page 796
NV_ROM_BAN- NER_TIMEOUT	0x210	EXP_RO M		0	See Table 608, “NV_ROM_BANNER_TIMEOUT Lay- out,” on page 799
NV_ROM_- BOOT_CONF1	0x221	EXP_RO M		1	See Table 596, “NV_ROM_BOOT_CONF1 Layout,” on page 786
NV_ROM_IB_- BOOT_CONF	0x222	EXP_RO M		0	See Table 598, “NV_ROM_IB_BOOT_CONF Layout,” on page 789
NV_ROM_IB_D HCP_CONF	0x223	EXP_RO M		0	See Table 502, “NV_ROM_IB_DHCP_CONF Layout,” on page 731
NV_ROM_ISC- SI_GENERAL_- CONF	0x230	EXP_RO M		2	See Table 600, “NV_ROM_ISCSI_GENERAL_CONF Layout,” on page 789
NV_ROM_ISC- SI_DHCP_VEN- DOR_ID	0x231	EXP_RO M			See Table 614, “NV_ROM_ISCSI_DHCP_VEN- DOR_ID Layout,” on page 801
NV_ROM_ISC- SI_INITIA- TOR_IP_ADDR	0x232	EXP_RO M			See Table 616, “NV_ROM_ISCSI_INITIA- TOR_IP_ADDR Layout,” on page 801
NV_ROM_ISC- SI_INITIATOR_- SUBNET_MASK	0x233	EXP_RO M			See Table 618, “NV_ROM_ISCSI_INITIATOR_SUB- NET_MASK Layout,” on page 802
NV_ROM_ISC- SI_INITIA- TOR_GATEWAY	0x234	EXP_RO M			See Table 620, “NV_ROM_ISCSI_INITIATOR_GATE- WAY Layout,” on page 802

Table 373 - Per-Host/Function Parameters [class_type=3]

Parameter Name	Index	Target	Cap	Ver sion	Reference
NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR	0x235	EXP_ROM			See Table 622, “NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR Layout,” on page 803
NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR	0x236	EXP_ROM			See Table 624, “NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR Layout,” on page 803
NV_ROM_ISCSI_INITIATOR_NAME	0x237	EXP_ROM			See Table 626, “NV_ROM_ISCSI_INITIATOR_NAME Layout,” on page 804
NV_ROM_ISCSI_INITIATOR_CHAP_ID	0x238	EXP_ROM			See Table 628, “NV_ROM_ISCSI_INITIATOR_CHAP_ID Layout,” on page 804
NV_ROM_ISCSI_INITIATOR_CHAP_PWD	0x239	EXP_ROM			See Table 630, “NV_ROM_ISCSI_INITIATOR_CHAP_PWD Layout,” on page 805
NV_ROM_ISCSI_DHCP_CONF	0x23A	EXP_ROM		0	See Table 612, “NV_ROM_ISCSI_DHCP_CONF Layout,” on page 800
NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET	0x240	EXP_ROM			See Table 632, “NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET Layout,” on page 805
NV_ROM_ISCSI_FIRST_TARGET_ADDR	0x241	EXP_ROM			See Table 636, “NV_ROM_ISCSI_FIRST_TARGET_ADDR Layout,” on page 806
NV_ROM_ISCSI_FIRST_TARGET_PORT	0x242	EXP_ROM			See Table 640, “NV_ROM_ISCSI_FIRST_TARGET_PORT Layout,” on page 807
NV_ROM_ISCSI_FIRST_TARGET_LUN	0x243	EXP_ROM			See Table 644, “NV_ROM_ISCSI_FIRST_TARGET_LUN Layout,” on page 808
NV_ROM_ISCSI_FIRST_TARGET_NAME	0x244	EXP_ROM			See Table 648, “NV_ROM_ISCSI_FIRST_TARGET_NAME Layout,” on page 809
NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID	0x245	EXP_ROM			See Table 652, “NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID Layout,” on page 810

Table 373 - Per-Host/Function Parameters [class_type=3]

Parameter Name	Index	Target	Cap	Ver sion	Reference
NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD	0x247	EXP_ROM			See Table 656, “NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD Layout,” on page 811
NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET	0x250	EXP_ROM			See Table 634, “NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET Layout,” on page 806
NV_ROM_ISCSI_SECOND_TARGET_ADDR	0x251	EXP_ROM			See Table 638, “NV_ROM_ISCSI_SECOND_TARGET_ADDR Layout,” on page 806
NV_ROM_ISCSI_SECOND_TARGET_PORT	0x252	EXP_ROM			See Table 642, “NV_ROM_ISCSI_SECOND_TARGET_PORT Layout,” on page 807
NV_ROM_ISCSI_SECOND_TARGET_LUN	0x253	EXP_ROM			See Table 646, “NV_ROM_ISCSI_SECOND_TARGET_LUN Layout,” on page 808
NV_ROM_ISCSI_SECOND_TARGET_NAME	0x254	EXP_ROM			See Table 650, “NV_ROM_ISCSI_SECOND_TARGET_NAME Layout,” on page 809
NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID	0x255	EXP_ROM			See Table 654, “NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID Layout,” on page 810
NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD	0x257	EXP_ROM			See Table 658, “NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD Layout,” on page 811
<u>NV_PF_FW_INT_0</u>	<u>0x3F0</u>	<u>NIC_INTERNAL</u>		<u>0</u>	<u>Internal for FW use</u>
<u>NV_PF_FW_INT_1</u>	<u>0x3F1</u>	<u>NIC_INTERNAL</u>		<u>0</u>	<u>Internal for FW use</u>

11.6.4 File Parameters

Table 374 - FILE Parameters [class_type=6]

Parameter Name	Index	Target	Cap	Version	Reference
FILE_APPLICABLE_TO	0x2	NIC		0	See Table 534, “FILE_APPLICABLE_TO Layout,” on page 747
FILE_COMMENT	0x3	NIC		0	See Table 536, “FILE_COMMENT Layout,” on page 748
FILE_SIGNATURE	0x4	NIC		0	See Table 538, “FILE_SIGNATURE Layout,” on page 748
FILE_DBG_FW_TOKEN_ID	0x5	NIC		0	See Table 540, “FILE_DBG_FW_TOKEN_ID Layout,” on page 749
FILE_CS_TOKEN_ID	0x7	NIC		0	See Table 542, “FILE_CS_TOKEN_ID Layout,” on page 750
FILE_MAC_ADDR_LIST	0x9	NIC		0	See Table 546, “FILE_MAC_ADDR_LIST Layout,” on page 751
FILE_PUBLIC_KEY	0xA	NIC		0	See Table 544, “FILE_PUBLIC_KEY Layout,” on page 750
FILE_SIGNATURE_4096_A	0xB	NIC		0	See Table 548, “FILE_SIGNATURE_4096_A Layout,” on page 752
FILE_SIGNATURE_4096_B	0xC	NIC		0	See Table 550, “FILE_SIGNATURE_4096_B Layout,” on page 753
FILE_PUBLIC_KEY_4096_A	0xD	NIC		0	See Table 554, “FILE_PUBLIC_KEY_4096_A Layout,” on page 756 [Unimplemented]
FILE_PUBLIC_KEY_4096_B	0xE	NIC		0	See Table 556, “FILE_PUBLIC_KEY_4096_B Layout,” on page 758 [Unimplemented]
FILE_PUBLIC_KEY_4096	0xF	NIC		0	See Table 552, “FILE_PUBLIC_KEY_4096 Layout,” on page 754

11.6.5 Per Host Parameters

Table 375 - Per-Host [class_type=7]

Parameter Name	Index	Target	Cap	Ver sion	Reference
	0x01				reserved for Safe mode internal TLV
NV_PCI_CONF	0x80	NIC		0	See Table 404, “NV_PCI_CONF Layout,” on page 668
NV_PCI_CAP	0x81	NIC	V	0	See Table 406, “NV_PCI_CAP Layout,” on page 670
SAFE_MODE_- CONF	0x82	NIC		0	See Table 408, “SAFE_MODE_CONF Layout,” on page 672
NV_HOST_PRIV_ CONF	0x83	NIC		0	See Table 410, “NV_HOST_PRIV_CONF Layout,” on page 673
NV_HOST_PRIV_ CAP	0x100	NIC	V	0	See Table 412, “NV_HOST_PRIV_CAP Layout,” on page 675
<u>NV_HOST_F- W_INT_0</u>	<u>0x3F0</u>	<u>NIC IN- TERNAL</u>		<u>0</u>	<u>Internal for FW use</u>

11.7 NIC Parameters

11.7.1 NV_BASE_MAC_GUID

Table 376 - NV_BASE_MAC_GUID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
base_mac[0]								base_mac[1]								base_mac[2]								base_mac[3]								00h
base_mac[4]								base_mac[5]								base_mac[6]								base_mac[7]								04h
base_guid[0]								base_guid[1]								base_guid[2]								base_guid[3]								08h
base_guid[4]								base_guid[5]								base_guid[6]								base_guid[7]								0Ch

Table 377 - NV_BASE_MAC_GUID Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	base_mac[8]	BYTES	The base MAC address of the device. The MAC addresses for the PCI physical functions of all the network ports are derived from the base MAC. Offset 00h, bits 15..0 hold bits 47..32 of the MAC address. Offset 04h, bits 31..0 hold bits 31..0 of the MAC address.
08h-0Ch	31:24	base_guid[8]	BYTES	The base InfiniBand GUID. The InfiniBand addresses for the PCI physical functions of all the network ports are derived from the base GUID.

11.7.2 NV_BASE_MAC_GUID_CAP

Table 378 - NV_BASE_MAC_GUID_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
num_of_allocated_guids																num_of_allocated_macs																04h

Table 379 - NV_BASE_MAC_GUID_CAP Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:16	num_of_allocated_guids	UNSIGNED	The total number of GUIDs that can be derived from this base GUID.
	15:0	num_of_allocated_macs	UNSIGNED	The total number of MAC addresses that can be derived from the base MAC.

11.7.3 NV_FORBIDDEN_VERSIONS

Table 380 - NV_FORBIDDEN_VERSIONS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset							
creation_time_year																creation_time_month								creation_time_day								00h							
creation_time_hour								creation_time_minute								creation_time_second																							04h
min_allowed_fw_version																																	08h						

Table 380 - NV_FORBIDDEN_VERSIONS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch
forbidden_fw_version[0]																																10h
forbidden_fw_version[1]																																14h
...																																...
forbidden_fw_version[31]																																8Ch

Table 381 - NV_FORBIDDEN_VERSIONS Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:16	creation_time_year	BINARY	The creation year of this FORBIDDEN_VERSION structure in 4 BCD digits.
	15:8	creation_time_month	BINARY	The creation month of this FORBIDDEN_VERSION structure in 2 BCD digits.
	7:0	creation_time_day	BINARY	The creation day of this FORBIDDEN_VERSION structure in 2 BCD digits. (GMT)
04h	31:24	creation_time_hour	BINARY	The creation hour of this FORBIDDEN_VERSION structure in 2 BCD digits. (GMT, 24h format)
	23:16	creation_time_minute	BINARY	The creation minute of this FORBIDDEN_VERSION structure in 2 BCD digits.
	15:8	creation_time_second	BINARY	The creation second of this FORBIDDEN_VERSION structure in 2 BCD digits.
08h	31:0	min_allowed_fw_version	BINARY	The minimal (earliest) firmware version priori to which the device must refuse update. Bits 31..24 - Major version bits 23..16: Minor version Bits 15..0 - Sub-minor version
10h-8Ch	31:0	forbidden_fw_version[32]	BINARY	A list of FW versions to which the device must refuse update. The FW version format is identical to the one shown in <i>min_allowed_fw_version</i> description. The list has up to 32 entries and is terminated by an entry set to 0x0.

11.7.4 NV_MEMIC_CONF

Table 382 - NV_MEMIC_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				memic_ - size_limit				log_memic_bar_size								00h

Table 383 - NV_MEMIC_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	11:8	memic_ - size_limit	ENUM	The maximum amount of internal device memory that can be consumed by the MEMIC application. 0x0: DISABLED 0x1: _256KB 0x2: _512KB 0x3: _1024KB mlxconfig_name: MEMIC_SIZE_LIMIT
	7:0	log_memic_bar_size	UNSIGNED	The amount of BAR size assigned for MEMIC. The size in bytes is <i>memic_size_limit</i> *2 ^{log_memic_bar_size} . mlxconfig_name: MEMIC_BAR_SIZE

11.7.5 NV_MEMIC_CAP

Table 384 - NV_MEMIC_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																				max_ - memic_ - size_limit				max_log_memic_bar_size								

Table 385 - NV_MEMIC_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	11:8	max_memic_size_limit	ENUM	The maximum value allowed for <i>memic_size_limit</i> . 0x0: DISABLED 0x1: _256KB 0x2: _512KB 0x3: _1024KB
	7:0	max_log_memic_bar_size	UNSIGNED	The maximal value allowed for <i>log_memic_bar_size</i>

11.7.6 NV_HOST_CHAINING_CONF

Table 386 - NV_HOST_CHAINING_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																											cache_disable		chain_mode			00h
log_num_descriptor[0]				log_num_descriptor[1]				log_num_descriptor[2]				log_num_descriptor[3]																				04h
log_num_descriptor[4]				log_num_descriptor[5]				log_num_descriptor[6]				log_num_descriptor[7]																				08h
log_data_size[0]				log_data_size[1]				log_data_size[2]				log_data_size[3]																				0Ch
log_data_size[4]				log_data_size[5]				log_data_size[6]				log_data_size[7]																				10h

Table 387 - NV_HOST_CHAINING_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	3	cache_disable	BOOL	<u>When TRUE, host chaining data is not cached on the device.</u> <u>mlxconfig_name: HOST_CHAINING_CACHE_DISABLE</u>
	1:0	chain_mode	ENUM	Enable and select host-chaining mode. 0x0: DISABLED 0x1: BASIC other values are reserved mlxconfig_name: HOST_CHAINING_MODE

Table 387 - NV_HOST_CHAINING_CONF Field Descriptions

Offset	Bits	Name	Type	Description
04h-08h	31:24	log_num_descriptor[8]	UNSIGNED	Log(base 2) of the number of packets descriptors that should be allocated by the host for host chaining for a given IEEE802.1p priority i. 0 means that no descriptors are allocated for this priority and traffic with this priority will be dropped. mlxconfig_name: HOST_CHAINING_DESCRIPTOR
0Ch-10h	31:24	log_data_size[8]	UNSIGNED	Log(base 2) of the buffer size (in bytes) allocated for host chaining for a given IEEE802.1p priority i. 0 means no buffer for this priority and traffic with this priority will be dropped. mlxconfig_name: HOST_CHAINING_TOTAL_BUFFER_SIZE

11.7.7 NV_HOST_CHAINING_CAP

Table 388 - NV_HOST_CHAINING_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
chain mode basic supported	cache_disable_supported																															00h
		log_max_total_num_descriptors								log_max_total_data_size								log_max_num_descriptor_per_prio								log_max_data_size_per_prio						

Table 389 - NV_HOST_CHAINING_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	chain_mode_basic_supported	BOOL	When set, <i>chain_mode</i> can be set to BASIC
	30	cache_disable_supported	BOOL	When TRUE, caching of host chaining data can be disabled by <u>cache_disable</u> in NV_HOST_CHAINING_CONF.

Table 389 - NV_HOST_CHAINING_CAP Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:24	log_max_total_num_descriptors	UNSIGNED	Log2 of the maximal total number of packet descriptors allocated for host chaining (sum of all priorities).
	23:16	log_max_total_data_size	UNSIGNED	Log(base 2) of the maximal total data buffer size allocated for host chaining (sum of all priorities)
	15:8	log_max_num_descriptor_per_prio	UNSIGNED	Log(base 2) of the maximal number of packet descriptors allocated for host chaining per priority
	7:0	log_max_data_size_per_prio	UNSIGNED	Log(base 2) of the maximal data buffer size allocated for host chaining per priority

11.7.8 NV_FPGA_CONF

Table 390 - NV_FPGA_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
auto_reload																																00h

Table 391 - NV_FPGA_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	auto_reload	BOOL	When set, the FPGA bitstream is reloaded from flash as part of the ConnectX boot sequence. mlxconfig_name: FPGA_AUTO_RELOAD

11.7.9 NV_FPGA_CAP

Table 392 - NV_FPGA_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
auto_reload_en																																00h

Table 393 - NV_FPGA_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	auto_reload_en	BOOL	When set, NV_FPGA_CONF.auto_rload is configurable

11.7.10 NV_INTERNAL_CPU_CONF

Table 394 - NV_INTERNAL_CPU_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												inter- nal_cpu_ - model	00h			
																													04h-0Ch			

Table 395 - NV_INTERNAL_CPU_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	3:0	internal_cpu_model	ENUM	Select the model for the Internal CPU 0x0: SEPERATED_HOST - Supported only when NV_INTERNAL_CPU_CAP.separate_host_model_supported==1. 0x1: EMBEDDED_CPU - Supported only when NV_INTERNAL_CPU_CAP.embedded_cpu_model_supported==1. other values are reserved This NVCONFIG parameter can only be configured from the embedded CPU. mlxconfig_name: INTERNAL_CPU_MODEL

11.7.11 NV_INTERNAL_CPU_CAP

Table 396 - NV_INTERNAL_CPU_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
<u>embedded_cpu_model_supported</u>																																00h
<u>separate_host_model_supported</u>																																04h-0Ch

Table 397 - NV_INTERNAL_CPU_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	separate_host_model_supported	BOOL	If set, SEPERATED_HOST model is supported for INTERNAL_CPU_MODEL.
	30	embedded_cpu_model_supported	BOOL	If set, EMBEDDED_CPU model is supported for INTERNAL_CPU_MODEL.

11.7.12 NV_FLEX_PARSER_CONF

Note that when this TLV is used, flex_ipv4_over_vxlan_end and flex_ipv4_over_vxlan_port in NV_SW_OFFLOAD_CONFIG can not be used.

Table 398 - NV_FLEX_PARSER_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																							flex_parser_profile_enable					00h				

Table 398 - NV_FLEX_PARSER_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																04h
																flex_ipv4_over_vxlan_port																08h
																																0Ch
																																10h

Table 399 - NV_FLEX_PARSER_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	7:0	flex_parser_profile_enable	UNSIGNED	Indicates which flex parser profile to enable. Each profile supports a set of protocols. The support indication and the set of protocols supported by profile 'x' reported in NC_FLEX_PARSER_CAP.flex_parser_profile_x_supported. mlxconfig_name: FLEX_PARSER_PROFILE_ENABLE
08h	15:0	flex_ipv4_over_vxlan_port	UNSIGNED	The UDP port for incoming IPoVxLAN traffic (non-standard). [Internal] This parameter is valid only when setting a profile that enabled flex_ipv4_over_vxlan. When set, also affects flex_vxlan_gpe_supported when enabled on same profile. mlxconfig_name: FLEX_IPV4_OVER_VXLAN_PORT

11.7.13 NV_FLEX_PARSER_CAP

Table 400 - NV_FLEX_PARSER_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
flex_parser_profile_0_supported																																
flex_parser_profile_1_supported																																
flex_parser_profile_2_supported																																
flex_parser_profile_3_supported																																

Table 400 - NV_FLEX_PARSER_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																04h-0Ch

Table 401 - NV_FLEX_PARSER_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	flex_parser_profile_0_supported	BOOL	If set, indicates profile 0 is supported and can be enabled by NV_FLEX_PARSER_CONF.flex_parser_profile_enable. Enabling this profile allows for parsing the following protocols: flex_geneve_supported flex_ipv4_over_vxlan_supported flex_ipv4_over_ip_supported flex_ipv6_over_ip_supported flex_vxlan_gpe_supported flex_geneve_tlv_option_0_supported
	30	flex_parser_profile_1_supported	BOOL	If set, indicates profile 1 is supported and can be enabled by NV_FLEX_PARSER_CONF.flex_parser_profile_enable. Enabling this profile allows for parsing the following protocols: flex_geneve_supported flex_ctrl_word_over_mpls_over_gre_supported flex_ctrl_word_over_mpls_over_udp_supported flex_geneve_tlv_option_0_supported
	29	flex_parser_profile_2_supported	BOOL	If set, indicates profile 2 is supported and can be enabled by NV_FLEX_PARSER_CONF.flex_parser_profile_enable. Enabling this profile allows for parsing the following protocols: flex_vxlan_gpe_supported flex_ipv4_over_vxlan_supported flex_icmp_supported flex_icmpv6_supported
	28	flex_parser_profile_3_supported	BOOL	If set, indicates profile 3 is supported and can be enabled by NV_FLEX_PARSER_CONF.flex_parser_profile_enable. Enabling this profile allows for parsing the following protocols: GTP-U

11.7.14 NV_WAKE_ON_LAN_CONF

The Wake on LAN parameter allows the user to enable/disable WOL on a per host-basis. Upon reception of a WOL packet with the host's MAC address, the NIC will waken the host if this parameter exists for, at least one of, the host's PCI functions.

Table 402 - NV_WAKE_ON_LAN_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
																					en_wol_magic													00h

Table 403 - NV_WAKE_ON_LAN_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	10	en_wol_magic	BOOL	Enables server Wake-on-LAN upon reception of WOL magic packet. [Internal] Support for this feature is indicated by the corresponding bit in the Wake on LAN Capabilities TLV mlxconfig_name: WOL_MAGIC_EN

11.7.15 NV_PCI_CONF

Table 404 - NV_PCI_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
advanced_pci_settings		force_eth_subclass		ibm_as_notify_en		exp_rom_tunneled_atomic_en		exp_rom_pxe_enable		exp_rom_uefi_arm_enable		exp_rom_nvme_uefi_x86_enable		exp_rom_nvme_uefi_arm_enable		exp_rom_nvme_uefi_x86_enable																00h

Table 405 - NV_PCI_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	advanced_pci_settings	BOOL	Show advanced PCI settings. mlxconfig_name: ADVANCED_PCI_SETTINGS
	30	ibm_capi_en	BOOL	Enable IBM's Coherent Accelerator Processor Interface (CAPI) mode. Supported only when ADVANCED_PCI_SETTINGS is set. mlxconfig_name: IBM_CAPI_EN
	29	force_eth_subclass	BOOL	Force the PCI function identifier with Ethernet subclass (00h). Supported only when ADVANCED_PCI_SETTINGS is set. mlxconfig_name: FORCE_ETH_PCI_SUBCLASS
	28	ibm_as_notify_en	BOOL	Enable IBM's AS Notify feature. Supported only when NV_PCI_CAP.ibm_as_notify_supported is set. mlxconfig_name: IBM_AS_NOTIFY_EN
	27	ibm_tunneled_atomic_en	BOOL	Enable IBM's Tunneled Atomic feature. Supported only when NV_PCI_CAP.ibm_tunneled_atomic_supported is set. mlxconfig_name: IBM_TUNNELED_ATOMIC_EN
	26	exp_rom_pxe_enable	BOOL	Enable Expansion ROM option for PXE mlxconfig_name: EXP_ROM_PXE_ENABLE
	25	exp_rom_uefi_x86_enable	BOOL	Enable Expansion ROM option for UEFI for x86 based host mlxconfig_name: EXP_ROM_UEFI_x86_ENABLE
	24	exp_rom_uefi_arm_enable	BOOL	Enable Expansion ROM option for UEFI for ARM based host mlxconfig_name: EXP_ROM_UEFI_ARM_ENABLE
	23	dynam-ic_vf_msix	BOOL	When set to TRUE, the device will support allocation of MSI-Xs to VFs by the PF, after SR-IOV is enabled (while the VF is down). mlxconfig_name: DYNAMIC_VF_MSIX_TABLE
	22	ats_enabled	BOOL	When set to TRUE, the device will support ATS. mlxconfig_name: ATS_ENABLED
	<u>21</u>	<u>exp_rom_nvme_uefi_x86_enable</u>	<u>BOOL</u>	<u>Enable Expansion ROM option for UEFI for x86 based host for NVMe functions</u> <u>mlxconfig_name: EXP_ROM_NVME_UEFI_x86_ENABLE</u>
	<u>20</u>	<u>exp_rom_nvme_uefi_arm_enable</u>	<u>BOOL</u>	<u>Enable Expansion ROM option for UEFI for ARM based host for NVMe functions</u> <u>mlxconfig_name: EXP_ROM_NVME_UEFI_ARM_ENABLE</u>
	<u>19</u>	<u>exp_rom_virtio_blk_uefi_x86_enable</u>	<u>BOOL</u>	<u>Enable Expansion ROM option for UEFI for x86 based host for VirtIO-BLK functions</u> <u>mlxconfig_name: EXP_ROM_VIRTIO_BLK_UEFI_x86_ENABLE</u>

Table 405 - NV_PCI_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	18	<u>exp_rom_virtio_blk_uefi_arm_enable</u>	BOOL	<u>Enable Expansion ROM option for UEFI for ARM based host for VirtIO-BLK functions</u> <u>mlxconfig_name: EXP_ROM_VIRTIO_BLK_UEFI_ARM_ENABLE</u>
	17	<u>exp_rom_virtio_net_uefi_x86_enable</u>	BOOL	<u>Enable Expansion ROM option for UEFI for x86 based host for VirtIO-NET functions</u> <u>mlxconfig_name: EXP_ROM_VIRTIO_NET_UEFI_x86_ENABLE</u>
	16	<u>exp_rom_virtio_net_uefi_arm_enable</u>	BOOL	<u>Enable Expansion ROM option for UEFI for ARM based host for VirtIO-NET functions</u> <u>mlxconfig_name: EXP_ROM_VIRTIO_NET_UEFI_ARM_ENABLE</u>
	15	<u>exp_rom_virtio_net_pxe_enable</u>	BOOL	<u>Enable Expansion ROM option for PXE for VirtIO-NET functions</u> <u>mlxconfig_name: EXP_ROM_VIRTIO_NET_PXE_ENABLE</u>

11.7.16 NV_PCI_CAP

Table 406 - NV_PCI_CAP Layout

Offset	00h
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	exp rom virtio net pxe supported
16	exp rom virtio net uefi arm supported
17	exp rom virtio net uefi x86 supported
18	exp rom virtio blk uefi arm supported
19	exp rom virtio blk uefi x86 supported
20	exp rom nvme uefi arm supported
21	exp rom nvme uefi x86 supported
22	ats supported
23	dynamic vf msix supported
24	exp rom uefi arm supported
25	exp rom uefi x86 supported
26	exp rom pxe supported
27	
28	ibm tunneled atomic supported
29	ibm as_notify supported
30	force eth subclass supported
31	ibm capi supported
	advanced pci_settings_supported

Table 407 - NV_PCI_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	advanced_pci_settings_supported	BOOL	When set, the PCI_FONF.advanced_pci_settings bit is configurable.
	30	ibm_capi_supported	BOOL	When set, the PCI_CONF.ibm_capi_en is supported.
	29	force_eth_subclass_supported	BOOL	When set, the PCI_CONF.force_eth_sub_class is supported.
	28	ibm_as_notify_supported	BOOL	When set, the PCI_CONF.ibm_as_notify_en is supported.
	27	ibm_tunneled_atomic_supported	BOOL	When set, the PCI_CONF.ibm_tunneled_atomic_en is supported.
	26	exp_rom_pxe_supported	BOOL	When set, NV_PCI_CONF.exp_rom_pxe_enable is supported.
	25	exp_rom_uefi_x86_supported	BOOL	When set NV_PCI_CONF.exp_rom_uefi_x86_enable is supported
	24	exp_rom_uefi_arm_supported	BOOL	When set, NV_PCI_CONF.exp_rom_uefi_arm_enable is supported.
	23	dynamic_vf_msix_supported	BOOL	When set to TRUE, setting NV_PCI_CONF.dynamic_vf_msix will be supported.
	22	ats_supported	BOOL	When set to TRUE, setting NV_PCI_CONF.ats_enabled will be supported.
	<u>21</u>	<u>exp_rom_nvme_uefi_x86_supported</u>	<u>BOOL</u>	<u>When TRUE, NV_PCI_CONF.exp_rom_nvme_uefi_x86_enable is supported</u>
	<u>20</u>	<u>exp_rom_nvme_uefi_arm_supported</u>	<u>BOOL</u>	<u>When TRUE, NV_PCI_CONF.exp_rom_nvme_uefi_arm_enable is supported</u>

Table 407 - NV_PCI_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	19	<u>exp_rom_virtio_blk_uefi_x86_supported</u>	BOOL	<u>When TRUE, NV_PCI_CONF.exp_rom_virtio_blk_uefi_x86_enable is supported</u>
	18	<u>exp_rom_virtio_blk_uefi_arm_supported</u>	BOOL	<u>When TRUE, NV_PCI_CONF.exp_rom_virtio_blk_uefi_arm_enable is supported</u>
	17	<u>exp_rom_virtio_net_uefi_x86_supported</u>	BOOL	<u>When TRUE, NV_PCI_CONF.exp_rom_virtio_net_uefi_x86_enable is supported</u>
	16	<u>exp_rom_virtio_net_uefi_arm_supported</u>	BOOL	<u>When TRUE, NV_PCI_CONF.exp_rom_virtio_net_uefi_arm_enable is supported</u>
	15	<u>exp_rom_virtio_net_pxe_supported</u>	BOOL	<u>When TRUE, NV_PCI_CONF.exp_rom_virtio_net_pxe_enable is supported</u>

11.7.17SAFE_MODE_CONF

Table 408 - SAFE_MODE_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
safe_mode_enable																									safe_mode_threshold												00h

Table 409 - SAFE_MODE_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	30	safe_ - mode_enable	BOOL	Enable Safe Mode activation after crossing a threshold of consecutive failed boots (no driver load). mlxconfig_name: SAFE_MODE_ENABLE
	7:0	safe_mode_ - threshold	UNSIGNED	Default value of the threshold for failed driver loads (driver load was not detected between reboots). When crossed, the device will load in Safe Mode with some default configurations, suspected as possible causes for driver load fail. mlxconfig_name: SAFE_MODE_THRESHOLD

11.7.18 NV_HOST_PRIV_CONF

Note: Marked as internal as there is a script for NV_HOST_PRIV_CONF as Raw TLV since users may be locked without unlock privileges.

Table 410 - NV_HOST_PRIV_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
			priv_level																									disable_rshim	disable_tracer	disable_counter_rd	disable_port_owner	00h
																																04h-0Ch

Table 411 - NV_HOST_PRIV_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	29:28	priv_level	ENUM	Privilege level of the host 0x0: PRIVILEGED - host has all supported privileges 0x1: LIMITED - host is not allowed to modify global/per port/parameters or access other hosts parameters [Internal] Limited privileges mean: - unable to access Global TLV and other host TLV - cannot perform reset - limited VSEC spaces - limited ICMDs mlxconfig_name: HOST_PRIV_LEVEL
	3	disable_rshim	BOOL	When TRUE, the host does not have an RSHIM function to access the embedded CPU registers mlxconfig_name: HOST_DISABLE_RSHIM
	2	disable_tracer	BOOL	When TRUE, the host will not be allowed to own the Tracer mlxconfig_name: HOST_DISABLE_TRACER_OWNER
	1	disable_counter_rd	BOOL	When TRUE, the host will not be allowed to read Physical port counters mlxconfig_name: HOST_DISABLE_PORT_COUNTER
	0	disable_port_owner	BOOL	When TRUE, the host will not be allowed to be Port Owner mlxconfig_name: HOST_DISABLE_PORT_OWNER

11.7.19 NV_HOST_PRIV_CAP

Note: Marked as internal as there is a script for NV_HOST_PRIV_CONF as Raw TLV since users may be locked without unlock privileges.

Table 412 - NV_HOST_PRIV_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
			priv_level_privileged																									disable_rshim_supported	disable_tracer_supported	disable_counter_rd_supported	disable_port_owner_supported	00h
			priv_level_limited																													04h-0Cch

Table 413 - NV_HOST_PRIV_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	29	priv_level_limited	BOOL	When TRUE, the host may be set to privilege level LIMITED
	28	priv_level_privileged	BOOL	When TRUE, the host may be set to privilege level PRIVILEGED
	3	disable_rshim_supported	BOOL	When TRUE, disable_rshim is supported in NV_HOST_PRIV_CONF
	2	disable_tracer_supported	BOOL	When TRUE, disable_tracer is supported in NV_HOST_PRIV_CONF
	1	disable_counter_rd_supported	BOOL	When TRUE, disable_counter_rd is supported in NV_HOST_PRIV_CONF
	0	disable_port_owner_supported	BOOL	When TRUE, disable_port_owner is supported in NV_HOST_PRIV_CONF

11.7.20 NV_WAKE_ON_LAN_CAP

Table 414 - NV_WAKE_ON_LAN_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset						
en_wol_magic																																						00h

Table 415 - NV_WAKE_ON_LAN_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	10	en_wol_magic	BOOL	This bit is set when Wake-On-Lan magic packet is supported for this host and port. Note that WOL support mandates a special circuitry on the NIC.

11.7.21 NV_VIRT_NET_ADDR

Table 416 - NV_VIRT_NET_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_mac_en		erase on powerup																										00h				
virtual_mac[0]								virtual_mac[1]								virtual_mac[2]								virtual_mac[3]							04h	
virtual_mac[4]								virtual_mac[5]								virtual_mac[6]								virtual_mac[7]							08h	

Table 417 - NV_VIRT_NET_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtual_mac_en	BOOL	When set, the PF will use the address defined in the 'virtual_mac' instead of the original MAC.
	29	erase_on_powerup	BOOL	When this bit is set, the MAC is ignored by firmware and the TLV is erased on powerup.

Table 417 - NV_VIRT_NET_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
04h-08h	31:24	virtual_mac[8]	BYTES	An alternate MAC for this PCI PF (48bits).

11.7.22 NV_VIRT_GUID_ADDR

Table 418 - NV_VIRT_GUID_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_guid_en		erase_on_powerup																										00h				
virtual_node_guid																											04h-08h					

Table 419 - NV_VIRT_GUID_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtual_guid_en	BOOL	When set, the PF will use the address defined in the 'virtual_node_guid' instead of the original MAC.
	29	erase_on_powerup	BOOL	When this bit is set, the GUID is ignored by firmware and the TLV is erased on powerup.
04h-08h	64	virtual_node_guid	BINARY	An alternate NODE GUID for this PCI PF (64 Bits).

11.7.23 NV_HCA_CONF

Table 420 - NV_HCA_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
multi_port_vhca_en		pci_data_wr_ordering_mode																														00h

Table 421 - NV_HCA_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	multi_port_vhca_en	BOOL	When set, the PF reports the multi-port-vhca capability. mlxconfig_name: MULTI_PORT_VHCA_EN
	29:28	pci_data_wr_ordering_mode	ENUM	Controls the PCI ordering attribute for write accesses from the device 0x0: per_mkey - Each MKey may be configured differently for Relaxed Ordering 0x1: force_relax - All MKeys will be set to use Relaxed Ordering at creation time show this setting only when NV_PCI_CONF.advanced_pci_settings is true. mlxconfig_name: PCI_WR_ORDERING

11.7.24 NV_HCA_CAP

Table 422 - NV_HCA_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
multi_port_vhca_supported			pci_data_wr_ordering_supported																													00h

Table 423 - NV_HCA_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	multi_port_vhca_supported	BOOL	When set NV_HCA_CONF. <i>multi_port_vhca_en</i> bit is configurable
	28	pci_data_wr_ordering_supported	BOOL	When set, NV_HCA_CONF pci_data_wr_ordering_mode is configurable

11.7.25 NV_ECPF_CONF

[Internal] This TLV is only available on embedded CPU PFs, FW should not indicate support for this TLV when accessed by the host.

Table 424 - NV_ECPF_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																						page_ supplier						esw_ manager	00h			
																																04h-0Ch

Table 425 - NV_ECPF_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	9:8	page_supplier	ENUM	Select who supplies ICM pages of the external host functions to device. [Internal] Note that ICM pages of ECPF are supplied always by the ECPF regardless of this configuration. 0x0: EXTERNAL_HOST_PF - External host supplies required ICM pages to handle it. 0x1: ECPF - ECPF supplies required ICM pages to handle the external host. mlxconfig_name: ECPF_PAGE_SUPPLIER
	1:0	esw_manager	ENUM	Select who owns the E-switch. 0x0: EXTERNAL_HOST_PF 0x1: ECPF mlxconfig_name: ECPF_ESWITCH_MANAGER

11.7.26NV_ECPF_CAP

Table 426 - NV_ECPF_CAP Layout

Offset	00h	
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		external host page supplier ecpf supported
29		external host page supplier external host pf supported
30		esw manager ecpf supported
31		esw manager external host pf supported

Table 426 - NV_ECPF_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																04h-0Ch

Table 427 - NV_ECPF_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	esw_man- ager_exter- nal_host_pf_s upported	BOOL	Set when the EXTERNAL_HOST_PF option is supported for ECPF_ESWITCH_MANAGER
	30	esw_man- ager_ecpf_- supported	BOOL	Set when the ECPF option is supported for ECP- F_ESWITCH_MANAGER
	29	exter- nal_host_page _supplier_exter- nal_host_p- f_supported	BOOL	Set when the EXTERNAL_HOST_PF option is supported for ECPF_PAGE_SUPPLIER
	28	exter- nal_host_page _supplier_ecp- f_supported	BOOL	Set when the ECPF option is supported for ECPF_PAGE_SUP- PLIER

11.7.27 NV_VPI_LINK_TYPE

Table 428 - NV_VPI_LINK_TYPE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												default _link_ - type			00h	

Table 431 - NV_GLOBAL_PCI_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	sriov_valid	BOOL	When set, the sriov field is valid. When cleared, the device uses the 'factory_settings' value.
	30	full_vf_qos_val id	BOOL	When set, the full_vf_qos field is valid. When cleared, the device uses the 'factory_settings' value.
	29	fpp_valid	BOOL	When set, the fpp field is valid. When cleared, the device uses the 'factory_settings' value.
	28	num_pfs_valid	BOOL	When set, the num_pfs field is valid. When cleared, the device uses the 'factory_settings' value.
	27	num_vf_msix- _valid	BOOL	When set, the num_vf_msix field is valid. When cleared, the device uses the 'factory_settings' value.
	26	num_pf_msix- _valid	BOOL	When set, the num_pf_msix field is valid. When cleared, the device uses the 'factory_settings' value.
	25	vf_bar_- size_valid	BOOL	When set, the vf_bar_size field is valid. When cleared, the device uses the 'factory_settings' value.
	24	pf_bar_- size_valid	BOOL	When set, the pf_bar_size field is valid. When cleared, the device uses the 'factory_settings' value.
	23	non_prefetch- able_p- f_bar_valid	BOOL	When set to TRUE, non_prefetchable_bar field and allow_large_non_prefetchable_pf_bar overrides the default value.
	22	vf_nodnic_en- able	BOOL	When set to TRUE, VF has VSC Gateway exposed through PCI, and may access the NODNIC initialization segment mlxconfig_name: VF_NODNIC_ENABLE
	21	strict_vf_m- six_num_en- abled	BOOL	When set to TRUE, num_vf_msix defines a strict number (no cal- culation and roundup). mlxconfig_name: STRICT_VF_MSIX_NUM
	18	vf_vpd_enable	BOOL	When set, VPD Capability is exposed to Virtual Functions. mlxconfig_name: VF_VPD_ENABLE
	17	non_prefetch- able_pf_bar	BOOL	When set, the PF BAR prefetchable bit is cleared. Note: PCI switches and operation systems have dedicated quotas for non-prefetchable memory, hence, you may need to decrease log_pf_bar_size to enable this feature. mlxconfig_name: NON_PREFETCHABLE_PF_BAR
	16	toolspf_en	BOOL	Add another PF (to each host) for debug tools. This bit is for Mel- lanox internal use only. The debug_tools PF is not associated to any port (and hence, has no MAC address).
	7:0	log_pf_bar2_- size	UNSIG NED	Log (base 2) of the size of a PF's BAR2 size, given in MB. mlxconfig_name: PF_BAR2_SIZE

Table 431 - NV_GLOBAL_PCI_CONF Field Descriptions

Offset	Bits	Name	Type	Description
04h	31	sriov_en	BOOL	Enable Single-Root I/O Virtualization (SR-IOV) mlxconfig_name: SRIOV_EN
	30	full_vf_qos	BOOL	When set, Virtual Function has the same number of traffic classes as physical functions.
	29	fpp_en	BOOL	When this bit is cleared, the device exposes a single PCI function for both ports. When set, the device exposes one or more PCI functions for each port (this is the only mode supported by ConnectX-4 devices). mlxconfig_name: FPP_EN
	28	hide_port2_pf	BOOL	When TRUE, the device will not advertise the PFs associated with port 2, except for the Embedded CPU (ECPF) if exists. This configuration is available only ~~if show this setting only~~ when NV_PCI_CONF.ADVANCED_PCI_SETTINGS is TRUE. mlxconfig_name: HIDE_PORT2_PF
	27	pf_bar2_en	BOOL	When set, BAR2 is exposed on all external host PFs (but not on the embedded ARM PFs/ECPFs). The BAR2 size is defined by the log_pf_bar2_size. mlxconfig_name: PF_BAR2_ENABLE
	23:16	num_pfs	UNSIGNED	Total number of PCIe functions (PFs) exposed by the device. The number of PFs must be a multiple of the number of ports, num_port_cap . Value 0 ~~indicates 256 PFs. [TBD Eitan]~~ is only supported when an Emulated PCI Switch is present.
	15:0	total_vfs	UNSIGNED	The total number of Virtual Functions (VFs) that can be supported, for each PF. mlxconfig_name: NUM_OF_VFS minval:1
08h	31:22	num_vf_msix	UNSIGNED	Number of MSI-X vectors and EQs per VF. mlxconfig_name: NUM_VF_MSIX
	21:12	num_pf_msix	UNSIGNED	Number of MSI-X vectors and EQs per PF. mlxconfig_name: NUM_PF_MSIX
	11:6	log_vf_uar_bar_size	UNSIGNED	Log 2 of the size of a VF's UAR BAR in MBs. mlxconfig_name: VF_LOG_BAR_SIZE
	5:0	log_pf_uar_bar_size	UNSIGNED	Log 2 of the size of a PF's UAR BAR in MBs. mlxconfig_name: PF_LOG_BAR_SIZE

Restrictions:

1) IF (fpp_en == 1) THEN (num_pfs)*(2^log_pf_uar_bar_size + 2^log_vf_uar_bar_size*total_vfs) <= max_total_bar ELSE
2^log_pf_uar_bar_sz + 2^log_vf_uar_bar_size*total_vfs <= max_total_bar

ELSE

$2^{\log_{\text{pf\_uar\_bar\_sz}}} + 2^{\log_{\text{vf\_uar\_bar\_size}} * \text{total\_vfs}} \leq \text{max\_total\_bar}$

$2) \text{num\_pfs} * (\text{num\_pf\_msix} + \text{total\_vfs} * \text{num\_vf\_msix}) \leq \text{max\_total\_msix}$

11.7.29 NV_GLOBAL_PCI_CAP

Table 432 - NV_GLOBAL_PCI_CAP Layout

Offset	00h	04h	08h	0Ch	10h
0		max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12		max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
13					
14					
15					
16					
17	max_acc_outstanding_read_byte_supported	max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
18	vf_vpd_supported				
19	pcie_credit_token_timeout_supported				
20					
21	strict_vf_msix_num_supported				
22	vf_nodnic_supported	max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
23	non_prefetchable_pf_bar_supported				
24	pf_bar_size_supported				
25	vf_bar_size_supported				
26	num_pf_msix_supported				
27	num_vf_msix_supported	max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
28	num_pfs_supported				
29	max_total_bar_valid				
30	max_total_msix_valid	max_vfs_per_pf	max_num_pfs	max_log_p- f_uar_bar_size	max_total_bar
31	max_vfs_per_pf_valid				

Table 433 - NV_GLOBAL_PCI_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	max-_vfs_per_pf_valid	BOOL	When set, the NIC will only respect NV_GLOBAL_PCI_CONF TLV if its total_vfs parameter is lower than or equal to the value of the max_vfs_per_pf field below.
	30	max_total_msix_valid	BOOL	When set, the NIC will only respect NV_GLOBAL_PCI_CONF TLV if the total MSIX required for all PFs and VFs is lower or equal to the number specified in the max_total_msix field below.
	29	max_total_bar_valid	BOOL	When set, the NIC will only respect an NV_GLOBAL_PCI_CONF TLV if the total bar size required for all PFs and VFs is lower or equal to the number specified in the max_total_bar field below.
	28	num_pfs_supported	BOOL	When set, the <i>num_pfs</i> field is configurable and <i>max_num_pfs</i> is valid.
	27	num_vf_msix_supported	BOOL	When set, the num_vf_msix field is configurable and the max_num_vf_msix field is valid.
	26	num_pf_msix_supported	BOOL	When set, the num_pf_msix field is configurable and the max_num_pf_msix field is valid.
	25	vf_bar_size_supported	BOOL	When set, the log_vf_uar_bar_size field is configurable and the max_log_vf_uar_bar_size field is valid.
	24	pf_bar_size_supported	BOOL	When set, the log_pf_uar_bar_size field is configurable and the max_log_pf_uar_bar_size field is valid.
	23	non_prefetchable_pf_bar_supported	BOOL	True when non_prefetchable_pf_bar field is configurable.
	22	vf_nodnic_supported	BOOL	When TRUE, the device supports enabling a VF to have VSC Gateway exposed through PCI, and may access the NODNIC initialization segment.
	21	strict_vf_msix_num_supported	BOOL	When set to TRUE, strict_vf_msix_num_enabled is valid.
	19	pcie_credit_token_timeout_supported	BOOL	When set, pcie_credit_token_timeout is supported.
	18	vf_vpd_supported	BOOL	When set, VPD Capability exposure to Virtual Functions is controlled by vf_vpd_enable.
	17	max_acc_outstanding_read_byte_supported	BOOL	When set, maximal accumulated outstanding read bytes can be limited by GLOBAL_PCI_CONF_2 max_acc_outstanding_read_byte.

Table 433 - NV_GLOBAL_PCI_CAP Field Descriptions

Offset	Bits	Name	Type	Description
04h	31	sriov_support	BOOL	When set, SR-IOV support is configurable by the NV_GLOBAL_PCI_CONF sriov_en bit.
	30	vf_qos_control_support	BOOL	When set, the device supports controlling the VF QoS setting.
	29	fpp_support	BOOL	When set, function per-port setting is configurable
	28	hide_port2_pf_supported	BOOL	When TRUE, hide_port2_pf is configurable
	27	pf_bar2_supported	BOOL	When set, exposing BAR2 to host PFs is supported.
	23:16	max_num_pfs	UNSIGNED	Maximum number of PFs. Value 0 indicates 256 PFs.
	15:0	max_vfs_per_pf	UNSIGNED	The maximum number of VFs that can be set in the total_vfs setting (per PF).
08h	31:22	max_num_vf_msix	UNSIGNED	Maximum number of MSI-X vectors and EQs per VF.
	21:12	max_num_pf_msix	UNSIGNED	Maximum number of MSI-X vectors and EQs per PF.
	11:6	max_log_vf_uar_bar_size	UNSIGNED	Log 2 of the maximum size (in MBs) of a VF's UAR BAR
	5:0	max_log_pf_uar_bar_size	UNSIGNED	Log 2 of the maximum size (in MBs) of a PF's UAR BAR
0Ch	31:0	max_total_msix	UNSIGNED	Maximum number of MSI-X for the aggregate of all PFs and VFs.
10h	31:0	max_total_bar	UNSIGNED	Maximum total of the MMIO space for all PFs and VFs combined, in megabytes. Note: if the number of PF or VF per port is not a power of 2, this parameter will also not be a power of 2.

11.7.30NV_GLOBAL_PCI_CAP_3

Table 434 - NV_GLOBAL_PCI_CAP_3 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bus0_restrict_en		bus0_width_max		bus0_aspm_en	bus0_speed_max										bus1_restrict_en	bus1_width_max		bus1_aspm_en	bus1_speed_max													00h
bus2_restrict_en		bus2_width_max		bus2_aspm_en	bus2_speed_max										bus3_restrict_en	bus3_width_max		bus3_aspm_en	bus3_speed_max													04h
bus4_restrict_en		bus4_width_max		bus4_aspm_en	bus4_speed_max										bus5_restrict_en	bus5_width_max		bus5_aspm_en	bus5_speed_max													08h
bus6_restrict_en		bus6_width_max		bus6_aspm_en	bus6_speed_max										bus7_restrict_en	bus7_width_max		bus7_aspm_en	bus7_speed_max													0Ch
																																10h-2Ch

Table 435 - NV_GLOBAL_PCI_CAP_3 Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	bus0_restrict_en	BOOL	When TRUE, PCI bus width, speed and ASPM can be restricted according to PCI_BUS_RESTRICT_WIDTH, PCI_BUS_RESTRICT_SPEED, PCI_BUS_RESTRICT_ASPM respectively
	30:28	bus0_width_max	ENUM	Maximal supported PCI bus width for restriction 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	27	bus0_aspm_en	BOOL	When TRUE, PCI bus ASPM may be restricted
	26:24	bus0_speed_max	ENUM	Maximal supported PCI speed width for restriction 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5
	15	bus1_restrict_en	BOOL	See bus0_restrict_en
	14:12	bus1_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	11	bus1_aspm_en	BOOL	See bus0_aspm_en
	10:8	bus1_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5

Table 435 - NV_GLOBAL_PCI_CAP_3 Field Descriptions

Offset	Bits	Name	Type	Description
04h	31	bus2_restrict_en	BOOL	See bus0_restrict_en
	30:28	bus2_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	27	bus2_aspm_en	BOOL	See bus0_aspm_en
	26:24	bus2_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5
	15	bus3_restrict_en	BOOL	See bus0_restrict_en
	14:12	bus3_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	11	bus3_aspm_en	BOOL	See bus0_aspm_en
	10:8	bus3_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5

Table 435 - NV_GLOBAL_PCI_CAP_3 Field Descriptions

Offset	Bits	Name	Type	Description
08h	31	bus4_re-strict_en	BOOL	See bus0_restrict_en
	30:28	bus4_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	27	bus4_aspm_en	BOOL	See bus0_aspm_en
	26:24	bus4_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5
	15	bus5_re-strict_en	BOOL	See bus0_restrict_en
	14:12	bus5_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	11	bus5_aspm_en	BOOL	See bus0_aspm_en
	10:8	bus5_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5

Table 435 - NV_GLOBAL_PCI_CAP_3 Field Descriptions

Offset	Bits	Name	Type	Description
0Ch	31	bus6_re-strict_en	BOOL	See bus0_restrict_en
	30:28	bus6_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	27	bus6_aspm_en	BOOL	See bus0_aspm_en
	26:24	bus6_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5
	15	bus7_re-strict_en	BOOL	See bus0_restrict_en
	14:12	bus7_width_max	ENUM	See bus0_width_max 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16
	11	bus7_aspm_en	BOOL	See bus0_aspm_en
	10:8	bus7_speed_max	ENUM	See bus0_speed_max 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5

11.7.31 NV_GLOBAL_PCI_CONF_2

Table 436 - NV_GLOBAL_PCI_CONF_2 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
max_acc_outstanding_read_byte																pcie_credit_token_timeout												00h				

Table 437 - NV_GLOBAL_PCI_CONF_2 Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:16	max_acc_outstanding_read_byte	UNSIGNED	Maximum accumulated outstanding Read requests bytes. Value is given in units of 1K bytes. The limit is applied for each PCI bus link individually. Value 0x0 indicates the use of device defaults. Configuration is available only when NV_PCI_CONF.ADVANCED_PCI_SETTINGS is TRUE. mlxconfig_name: MAX_ACC_OUT_READ
	15:0	pcie_credit_token_timeout	UNSIGNED	PCIe credit timeout. In case a pending transaction has no credits longer than this timeout value, buffered transactions will be dropped. Value is given in milliseconds in the range of 10 to 4,000. Value 0x0 indicates the device will adjust the timeout automatically. Value 0xffff disables the feature. [Internal]: Actual value is rounded down to closest 10 ms value] mlxconfig_name: PCIE_CREDIT_TOKEN_TIMEOUT

11.7.32 NV_GLOBAL_PCI_CONF_3

[Internal] This TLV is ultra-fast, FW always puts it first in the fast list so it could access it early in the boot stage. consider carefully which params should be added here.

Table 438 - NV_GLOBAL_PCI_CONF_3 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bus0_restrict	bus0_wi	bus0_aspm	bus0_speed												bus1_restrict	bus1_wi	bus1_aspm	bus1_speed														00h
bus2_restrict	bus2_wi	bus2_aspm	bus2_speed												bus3_restrict	bus3_wi	bus3_aspm	bus3_speed														04h
bus4_restrict	bus4_wi	bus4_aspm	bus4_speed												bus5_restrict	bus5_wi	bus5_aspm	bus5_speed														08h

Table 438 - NV_GLOBAL_PCI_CONF_3 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset															
bus6_restrict				bus6_wi dth				bus6_aspm				bus6_- speed								bus7_restrict				bus7_wi dth				bus7_aspm				bus7_- speed								0Ch				10h-2Ch			

Table 439 - NV_GLOBAL_PCI_CONF_3 Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	bus0_restrict	BOOL	When TRUE, PCI bus width, speed and ASPM will be restricted according to PCI_BUS_RESTRICT_WIDTH, PCI_BUS_RESTRICT_SPEED, PCI_BUS_RESTRICT_ASPM respectively mlxconfig_name: PCI_BUS0_RESTRICT
	30:28	bus0_width	ENUM	Restricts the PCI bus width to be smaller or equal to: 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 Valid when PCI_BUS_RESTRICT is TRUE. mlxconfig_name: PCI_BUS0_RESTRICT_WIDTH
	27	bus0_aspm	BOOL	When FALSE, PCI bus will not have ASPM enabled. Valid when PCI_BUS_RESTRICT is TRUE. mlxconfig_name: PCI_BUS0_RESTRICT_ASPM
	26:24	bus0_speed	ENUM	Restricts the PCI speed to be smaller or equal to: 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 Valid when PCI_BUS_RESTRICT is TRUE. mlxconfig_name: PCI_BUS0_RESTRICT_SPEED
	15	bus1_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS1_RESTRICT
	14:12	bus1_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS1_RESTRICT_WIDTH
	11	bus1_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS1_RESTRICT_ASPM
	10:8	bus1_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS1_RESTRICT_SPEED

Table 439 - NV_GLOBAL_PCI_CONF_3 Field Descriptions

Offset	Bits	Name	Type	Description
04h	31	bus2_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS2_RESTRICT
	30:28	bus2_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS2_RESTRICT_WIDTH
	27	bus2_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS2_RESTRICT_ASPM
	26:24	bus2_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS2_RESTRICT_SPEED
	15	bus3_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS3_RESTRICT
	14:12	bus3_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS3_RESTRICT_WIDTH
	11	bus3_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS3_RESTRICT_ASPM
	10:8	bus3_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS3_RESTRICT_SPEED

Table 439 - NV_GLOBAL_PCI_CONF_3 Field Descriptions

Offset	Bits	Name	Type	Description
08h	31	bus4_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS4_RESTRICT
	30:28	bus4_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS4_RESTRICT_WIDTH
	27	bus4_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS4_RESTRICT_ASPM
	26:24	bus4_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS4_RESTRICT_SPEED
	15	bus5_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS5_RESTRICT
	14:12	bus5_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS5_RESTRICT_WIDTH
	11	bus5_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS5_RESTRICT_ASPM
	10:8	bus5_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS5_RESTRICT_SPEED

Table 439 - NV_GLOBAL_PCI_CONF_3 Field Descriptions

Offset	Bits	Name	Type	Description
0Ch	31	bus6_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS6_RESTRICT
	30:28	bus6_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS6_RESTRICT_WIDTH
	27	bus6_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS6_RESTRICT_ASPM
	26:24	bus6_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS6_RESTRICT_SPEED
	15	bus7_restrict	BOOL	See PCI_BUS0_RESTRICT mlxconfig_name: PCI_BUS7_RESTRICT
	14:12	bus7_width	ENUM	See PCI_BUS0_RESTRICT_WIDTH 0x0: PCI_X1 0x1: PCI_X2 0x2: PCI_X4 0x3: PCI_X8 0x4: PCI_X16 mlxconfig_name: PCI_BUS7_RESTRICT_WIDTH
	11	bus7_aspm	BOOL	See PCI_BUS0_RESTRICT_ASPM mlxconfig_name: PCI_BUS7_RESTRICT_ASPM
	10:8	bus7_speed	ENUM	See PCI_BUS0_RESTRICT_SPEED 0x0: PCI_GEN_1 0x1: PCI_GEN_2 0x2: PCI_GEN_3 0x3: PCI_GEN_4 0x4: PCI_GEN_5 mlxconfig_name: PCI_BUS7_RESTRICT_SPEED

11.7.33 NV_TPT

Table 440 - NV_TPT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												log_max_pay- load_size	00h			

Table 441 - NV_TPT Field Descriptions

Offset	Bits	Name	Type	Description
00h	3:0	log_max_payload_size	ENUM	Set the PCIe burst size for the NIC internal Translation and Protection (TPT) mechanism. 0x0: AUTOMATIC 0xC: _4KB mlxconfig_name: INT_LOG_MAX_PAYLOAD_SIZE

11.7.34 NV_TPT_CONF

Table 442 - NV_TPT_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												log_max_pay- load_size	00h			

Table 443 - NV_TPT_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	3:0	log_max_payload_size	ENUM	Sets the PCIe burst size for the NIC internal Translation and Protection (TPT) mechanism. 0x0: AUTOMATIC 0xC: _4KB mlxconfig_name: INT_LOG_MAX_PAYLOAD_SIZE

Table 445 - NV_SW_OFFLOAD_CONFIG Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:16	ip_over_vx-lan_port	UNSIGNED	This parameter is only supported if NV_SW_OFFLOAD_CAP. <i>ip_over_vxlan_supported</i> is set. mlxconfig_name: IP_OVER_VXLAN_PORT mlxconfig_desc: The UDP port for incoming IPoVxLAN traffic (non-standard). minval:1
	15	tunnel_ecn_copy_offload_disable	BOOL	When TRUE, ECN field copy from inner to outer header during encapsulation is disabled. mlxconfig_name: TUNNEL_ECN_COPY_DISABLE
	14:12	pci_atomic_mode	ENUM	PCI Atomic mode. 0x0: PCI_ATOMIC_DISABLED_EXT_ATOMIC_ENABLED. 0x1: PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_SERIALIZED - Atomic serialization will guarantee atomicity for multiple accesses to the same address supported only when NV_PCI_CAP.PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_SERIALIZED==1 . 0x2: PCI_ATOMICS_ENABLED_EXT_ATOMICS_DISABLED - supported only when NV_PCI_CAP.PCI_ATOMICS_ENABLED_EXT_ATOMICS_DISABLED . 0x3: PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_NONCOHERENT - supported only when NV_PCI_CAP.PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_NONCOHERENT . 0x4: PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED - supported only when NV_PCI_CAP.PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED . mlxconfig_name: PCI_ATOMIC_MODE
	<u>9</u>	<u>vector_calc_disable</u>	<u>BOOL</u>	<u>When TRUE, Vector-Calc operations are disabled</u> <u>mlxconfig_name: VECTOR_CALC_DISABLE</u>
	8	uctx_en	BOOL	When set to TRUE, the device supports creation of UCTX contexts. mlxconfig_name: UCTX_EN [Internal] allow devx
	7	prio_tag_required_en	BOOL	When set to TRUE, the system requires to always have prio tag on the network. SW of e-Switch manager should set the required flow table rules to ensure that. This field is supported only when NV_SW_OFFLOAD_CAP.prio_tag_required==1 and is reported in HCA_CAP.prio_tag_required. mlxconfig_name: PRIO_TAG_REQUIRED_EN [Internal] This field is added as a solution for verizon where Pop vlan not supported in Tx and push not supported in Rx.

Table 445 - NV_SW_OFFLOAD_CONFIG Field Descriptions

Offset	Bits	Name	Type	Description
00h	6	esw_fdb_ipv4_ttl_modify_enable	BOOL	When TRUE, the device will supports e-switch rules modifying TTL of packets from the uplink to a vport. mlxconfig_name: ESWITCH_IPV4_TTL_MODIFY_ENABLE
00h	5	mkey_by_name	BOOL	When TRUE, the device supports allocating MKey numbers by the device driver [Internal] default must be 0x0 mlxconfig_name: MKEY_BY_NAME
	4	ip_over_vxlan_en	BOOL	This parameter is only supported if NV_SW_OFFLOAD_CAP. ip_over_vxlan_supported is set. mlxconfig_name: IP_OVER_VXLAN_EN mlxconfig_desc: When set, non-standard IPoVxLAN offload is enabled for incoming UDP port specified in IP_OVER_VXLAN_PORT.
	3	one_qp_per_recovery	BOOL	When set, a lost doorbell will trigger WQ doorbell record query only for the QP ringing the doorbell. When cleared, a lost doorbell may trigger WQ doorbell record query for multiple QPs supported_from_version:1
	2:0	cqe_compression	ENUM	Configure which algorithm should be used by the NIC in order to decide when to activate CQE compression based on PCIe bus condition. Note that the driver can enable compression on a per CQE basis 0x0: BALANCED 0x1: AGGRESSIVE mlxconfig_name: CQE_COMPRESSION

Table 445 - NV_SW_OFFLOAD_CONFIG Field Descriptions

Offset	Bits	Name	Type	Description
04h	<u>28:24</u>	<u>log_max_outstanding_wqe</u>	<u>UNSIG NED</u>	<u>Log2 of the maximal amount of outstanding (uncompleted) WQEs a single Transmit Work Queue may hold. Additional WQEs will be delayed until some of the outstanding WQEs are completed.</u> <u>Value 0x0 indicates device default.</u> <u>Value 0x1F indicates device maximal supported value</u> <u>mlxconfig_name: LOG_MAX_OUTSTANDING_WQE</u>
	15:12	lro_log_time-out3	UNSIG NED	Log2 of Large Receive Offload (LRO) timeout #3, in microseconds. Driver can select one of the 4 configured LRO timeouts on a per Qp basis in run-time (lro_timeout_period_usecs field of the TIR context) mlxconfig_name: LRO_LOG_TIMEOUT3
	11:8	lro_log_time-out2	UNSIG NED	Log2 of Large Receive Offload (LRO) timeout #2, in microseconds. Driver can select one of the 4 configured LRO timeouts on a per Qp basis in run-time (lro_timeout_period_usecs field of the TIR context) mlxconfig_name: LRO_LOG_TIMEOUT2
	7:4	lro_log_time-out1	UNSIG NED	Log2 of Large Receive Offload (LRO) timeout #1, in microseconds. Driver can select one of the 4 configured LRO timeouts on a per Qp basis in run-time (lro_timeout_period_usecs field of the TIR context) mlxconfig_name: LRO_LOG_TIMEOUT1
	3:0	lro_log_time-out0	UNSIG NED	Log2 of Large Receive Offload (LRO) timeout #0, in microseconds. Driver can select one of the 4 configured LRO timeouts on a per Qp basis in run-time (lro_timeout_period_usecs field of the TIR context) mlxconfig_name: LRO_LOG_TIMEOUT0

11.7.36NV_SW_OFFLOAD_CAP

Table 446 - NV_SW_OFFLOAD_CAP Layout

Offset	00h
0	
1	
2	
3	
4	
5	
6	
7	prio_tag_required_supported
8	uctx_supported
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	max_outstanding_wqe_supported
21	vector_calc_disable_supported
22	tunnel_ecn_copy_offload_disable_supported
23	esw_fdb_ipv4_ttl_modify_supported
24	pci_atomics_disabled_ext_atomics_enabled
25	pci_atomics_enabled_ext_atomics_enabled_serialized
26	pci_atomics_enabled_ext_atomics_disabled
27	pci_atomics_enabled_ext_atomics_enabled_nocohere
28	pci_atomics_enabled_ext_atomics_enabled
29	
30	mkey_by_name_supported
31	ip_over_vxlan_supported

Table 447 - NV_SW_OFFLOAD_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	ip_over_vxlan_supported	BOOL	When set, both <i>ip_over_vxlan_en</i> and <i>ip_over_vxlan_port</i> in NVSW_OFFLOAD_CONFIG are configurable.
	30	mkey_by_name_supported	BOOL	When TRUE, NV_HCA_CONF mkey_by_name is configurable
	28	pci_atomics_enabled_ext_atomics_enabled	BOOL	When set, NV_HCA_CONF.pci_atomic_mode supports PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED mode.
	27	pci_atomics_enabled_ext_atomics_enabled_nocoherent	BOOL	When set, NV_HCA_CONF.pci_atomic_mode supports PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_NONCOHERENT mode.
	26	pci_atomics_enabled_ext_atomics_disabled	BOOL	When set, NV_HCA_CONF.pci_atomic_mode supports PCI_ATOMICS_ENABLED_EXT_ATOMICS_DISABLED mode.
	25	pci_atomics_enabled_ext_atomics_enabled_serialized	BOOL	When set, NV_HCA_CONF.pci_atomic_mode supports PCI_ATOMICS_ENABLED_EXT_ATOMICS_ENABLED_SERIALIZED mode.
	24	pci_atomics_disabled_ext_atomics_enabled	BOOL	When set, NV_HCA_CONF.pci_atomic_mode supports PCI_ATOMICS_DISABLED_EXT_ATOMICS_ENABLED mode.
	23	esw_fdb_ipv4_ttl_modify_supported	BOOL	When set to TRUE, the device supports enabling e-switch rules that modify TTL of packets from the uplink to a vport, using ESWITCH_IPV4_TTL_MODIFY_ENABLED.
	22	tunnel_ecn_copy_offload_disable_supported	BOOL	When TRUE, tunnel_ecn_copy_offload_disable in NV_SW_OFFLOAD_CONFIG is supported.
	21	vector_calc_disable_supported	BOOL	When TRUE, Vector Calc operations can be disabled using <i>vector_calc_disable</i> on NV_SW_OFFLOAD_CONFIG
	20	max_outstanding_wqe_supported	BOOL	When TRUE, the maximal number of outstanding WQEs for a single Transmit Work Queue can be configured using <i>in NV_SW_OFFLOAD_CONFIG</i>

Table 447 - NV_SW_OFFLOAD_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	7	prio_tag_required_supported	BOOL	When set to TRUE, PRIO_TAG_REQUIRED_EN is supported.

11.7.37 NV_PACKET_PACING

Table 448 - NV_PACKET_PACING Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 449 - NV_PACKET_PACING Field Descriptions

Offset	Bits	Name	Type	Description
00h	1	lag_disable	BOOL	Disable LAG support
	0	packet_pacing	BOOL	Enable packet pacing

11.7.38 NV_FILE_ID_MLNX

Table 450 - NV_FILE_ID_MLNX Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 450 - NV_FILE_ID_MLNX Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hour								minutes								seconds																0Ch

Table 451 - NV_FILE_ID_MLNX Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	erase_low- er_priority	BOOL	When set, installation of the file will erase all NVCONFIG parameters that are at lower priority. If the file priority is MLNX, installation of the file will not copy any TLVs from the current NVDAT partition. If the file priority is VEND- OR_NVCONF, the installation of the file will only copy MLNX level TLVs.
04h	31:0	file_version	UNSIGN ED	32bit version number of the file.
08h	31:16	year	BINARY	year - 4 packed BCD. Note: If year is zero, FW may copy the signature time to this TLV before copying it to flash, so the NVCONFIG component will have a date&time prop- erty.
	15:8	month	BINARY	month - 2 packed BCD
	7:0	day	BINARY	day - 2 packed BCD
0Ch	31:24	hour	BINARY	hour - 2 packed BCD (UTC)
	23:16	minutes	BINARY	minutes - 2 packed BCD
	15:8	seconds	BINARY	seconds - 2 packed BCD

11.7.39NV_FILE_ID_VENDOR

Table 452 - NV_FILE_ID_VENDOR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
erase_low- er_priority																																00h
		file_version																														04h

Table 452 - NV_FILE_ID_VENDOR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
year																month						day						08h				
hour								minutes								seconds												0Ch				

Table 453 - NV_FILE_ID_VENDOR Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	erase_low- er_priority	BOOL	When set, installation of the file will erase all NVCONFIG parameters that are at lower priority. If the file priority is MLNX, installation of the file will not copy any TLVs from the current NVDAT partition. If the file priority is VENDOR_NVCONF, the installation of the file will only copy MLNX level TLVs.
04h	31:0	file_version	UNSIGNED	32bit version number of the file.
08h	31:16	year	BINARY	year - 4 packed BCD. Note: If year is zero, FW may copy the signature time to this TLV before copying it to flash , so the NVCONFIG component will have a date&time property.
	15:8	month	BINARY	month - 2 packed BCD
	7:0	day	BINARY	day - 2 packed BCD
0Ch	31:24	hour	BINARY	hour - 2 packed BCD (UTC)
	23:16	minutes	BINARY	minutes - 2 packed BCD
	15:8	seconds	BINARY	seconds - 2 packed BCD

11.7.40NV_PROG_CC

Table 454 - NV_PROG_CC Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
user_cc_en																																00h

Table 455 - NV_PROG_CC Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	user_cc_en	BOOL	When set to TRUE, the device will support programming user defined CC algorithm. mlxconfig_name: USER_PROGRAMMABLE_CC

11.7.41 NV_HPCC_INT

Table 456 - NV_HPCC_INT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hpcc_int_en		<u>np_rtt_int_data_mode</u>							<u>np_rtt_response_desc_valid</u>	<u>np_rtt_response_desc</u>																						00h
<u>system_rtt</u>																																04h
																																08h
																																0Ch

Table 457 - NV_HPCC_INT Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	hpcc_int_en	BOOL	When set to TRUE, the device will support HPCC CC based on INT stamped RTT packets. The HPCC algorithm should be programmed/configured separately to this configuration. mlxconfig_name: HPCC_INT_EN
	30:28	np_rtt_int_data_mode	ENUM	Defines how Notification Point (NP) generates the data for the RTT response, out of the telemetry (INT) information. 0x0: LAST_HOP - use the last telemetry entry in the packet 0x1: MOST_CONGESTED - use the telemetry entry indicating the most congested telemetry point. mlxconfig_name: HPCC_INT_NP_RTT_DATA_MODE
	22	np_rtt_response_dscp_valid	BOOL	When TRUE, Notification Point (NP) will generate RTT response using the DSCP value given in np_rtt_response_dscp When FALSE, RTT response will use the DSCP value from the RTT request mlxconfig_name: HPCC_INT_NP_RTT_DSCP_EN
	21:16	np_rtt_response_dscp	UNSIGNED	DSCP value used by Notification Point (NP) when generating RTT response. Valid only if np_rtt_response_dscp_valid is set. mlxconfig_name: HPCC_INT_NP_RTT_DSCP
04h	31:0	system_rtt	UNSIGNED	Estimation of the system Round Trip Time (RTT), given in nanoseconds. Valid only when np_rtt_int_data_mode is set to MOST_CONGESTED. Value 0x0 indicates using device defaults. mlxconfig_name: HPCC_INT_SYSTEM_RTT

11.7.42 NV_HPCC_INT_CAP

Table 458 - NV_HPCC_INT_CAP Layout

Offset	00h	04h	08h	0Ch
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28	system_rtt_supported			
29	np_rtt_response_dscp_supported			
30	np_rtt_int_data_mode_most_congested			
31	np_rtt_int_data_mode_last_hop			

Table 459 - NV_HPCC_INT_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	np_rtt_int_data_mode_last_hop	BOOL	When TRUE, <u>np_rtt_int_data_mode</u> in NV_HPCC_INT may be set to LAST_HOP
	30	np_rtt_int_data_mode_most_congested	BOOL	When TRUE, <u>np_rtt_int_data_mode</u> in NV_HPCC_INT may be set to MOST_CONGESTED
	29	np_rtt_response_dscp_supported	BOOL	When TRUE, <u>np_rtt_response_dscp_valid</u> in NV_HPCC_INT may be set to TRUE, and the DSCP value used for NP RTT responses can be set in <u>np_rtt_response_dscp</u>
	28	system_rtt_supported	BOOL	When TRUE, <u>system_rtt</u> in NV_HPCC_INT may be set.

11.7.43 NV_IB_DC_CONF

Table 460 - NV_IB_DC_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																											log_d-cr_hash_table_size					00h
								dcr_lifo_size																				04h				

Table 461 - NV_IB_DC_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	4:0	log_d-cr_hash_table_size	UNSIGNED	InfiniBand DC (Dynamically Connected) hash table log2 size (for example: 7 means 128 entries) mlxconfig_name: LOG_DCR_HASH_TABLE_SIZE
04h	23:0	dcrlifo_size	UNSIGNED	The total amount of InfiniBand DCRs (Dynamically Connected Responders), minus one. mlxconfig_name: DCR_LIFO_SIZE

11.7.44 NV_IB_DC_CAP

Table 462 - NV_IB_DC_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
min_log_dcr_hash_table_size																																00h
max_log_dcr_hash_table_size																																04h
min_dcr_lifo_size																																08h
max_dcr_lifo_size																																0Ch

Table 463 - NV_IB_DC_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:0	min_log_d-cr_hash_table_size	UNSIGNED	Minimal value supported for log_dcr_hash_table_size

Table 463 - NV_IB_DC_CAP Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:0	max_log_dcr_hash_table_size	UNSIGNED	Maximal value supported for log_dcr_hash_table_size
08h	31:0	min_dcr_lifo_size	UNSIGNED	Minimal value supported for dcr_lifo_size
0Ch	31:0	max_dcr_lifo_size	UNSIGNED	Maximal value supported for dcr_lifo_size
10h	31:0	min_dcr_cnack_buffer_size	UNSIGNED	The minimum configurable value for dcr_cnack_buffer_size.
14h	31:0	max_dcr_cnack_buffer_size	UNSIGNED	The maximum configurable value for dcr_cnack_buffer_size.

11.7.45 NV_ENCLOSURE_INFO

Table 464 - NV_ENCLOSURE_INFO Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																hp_7kd_en	00h
bay_number								tray_location								tray_id								chassis_id								04h	

Table 465 - NV_ENCLOSURE_INFO Field Descriptions

Offset	Bits	Name	Type	Description
00h	0	hp_7kd_en	BOOL	When this bit is set to 1, the NIC PHY should apply PHY calibration according to the data specified in the following fields

Table 465 - NV_ENCLOSURE_INFO Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:24	bay_number	UNSIGNED	
	23:16	tray_location	ENUM	0x0: SINGLE_NODE_TRAY 0x1: FRONT 0x2: REAR
	15:8	tray_id	UNSIGNED	
	7:0	chassis_id	UNSIGNED	

11.7.46 NV_CURRENT_CONFIGURATOR

Table 466 - NV_CURRENT_CONFIGURATOR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								current_cfg_method				00h				

Table 467 - NV_CURRENT_CONFIGURATOR Field Descriptions

Offset	Bits	Name	Type	Description
00h	7:0	current_cfg_method	ENUM	Parameter from "HP ASN1 Data Structure Specification v2.8.6.asn" 0x0: E_NONE - default config 0x1: OS - O/S making cfg changes in the absences of any other Mgmt Util (command interface) 0x2: E_VENDOR_SPEC - e.g. Vendor's Option ROM, NPAR, O/S Cfg Utils (FW writes this TLV when a TLV is written from mlxconfig, that is when NVDA write for any TLV is coming from the ICMD gateway without a UEFI or flexboot 'writer-ID'). 0x3: E_HP_OTHER - (when an extern-DCi-mgmt parameter with value e-None is arriving from the BMC ASN.1 API) 0x4: E_VC_CLP - Driver or F/W detects HPE Classic C-Class SM-CLP 0x5: E_HP_CNU - Driver or F/W detects HPE's Converged Network Utility (when an extern-DCi-mgmt parameter with value <to-be-defined-by-HP> is arriving from the BMC ASN.1 API) 0x6: E_HP_ONEVIEW - If manage-via-DCi value equals to HPE-OneView(6) in BaseDev-Cfg-Now is coming from the BMC ASN.1 API 0x7: E_UEFI_HII - UEFI Option ROM executing

11.7.47NV_INVALIDATE_ALL_TIMESTAMP

Table 468 - NV_INVALIDATE_ALL_TIMESTAMP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
time_synced																												00h				
																	time											04h-08h				

Table 469 - NV_INVALIDATE_ALL_TIMESTAMP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	time_synced	BOOL	When set, the time field represents the number of microseconds passed since 1/1/1970. Otherwise, time is the number of microseconds passed since the NIC boot (uptime)
04h-08	64	time	UNSIGNED	Time in microseconds

11.7.48 NV_DEBUG_MODE

Table 470 - NV_DEBUG_MODE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																											force_pf_rate_limiter		nvia_nvlog_on_boot	nvia_nvconfig_on_boot	00h	

Table 471 - NV_DEBUG_MODE Field Descriptions

Offset	Bits	Name	Type	Description
00h	3	force_pf_rate_limiter	BOOL	Qos mode will be forced as pf rate limiter. [internal] This field is used for verification purposes only. We will add assert in fw in case it active together with packet pacing.
	1	nvia_nvlog_on_boot	BOOL	Debug only. When set, firmware erases the NVLOG partition on BOOT and erases this NVCONFIG TLV.
	0	nvia_nvconfig_on_boot	BOOL	Debug only. When set, firmware invalidates all NVCONFIG parameters on boot including this parameter.

11.7.49 NV_LLDP_NB_DCBX

Table 472 - NV_LLDP_NB_DCBX Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																													dcbx_willing	cee_dcbx_en	ieee_dcbx_en	00h

Table 473 - NV_LLDP_NB_DCBX Field Descriptions

Offset	Bits	Name	Type	Description
00h	2	dcbx_willing	BOOL	Allow the NIC to accept DCBX configuration from the remote peer. mlxconfig_name: DCBX_WILLING
	1	cee_dcbx_en	BOOL	Enable DCBX in CEE mode mlxconfig_name: DCBX_CEE
	0	ieee_dcbx_en	BOOL	Enable DCBX in IEEE mode mlxconfig_name: DCBX_IEEE

11.7.50 NV_LLDP_NB_TX_CHASSIS_ID_TLV

Table 474 - NV_LLDP_NB_TX_CHASSIS_ID_TLV Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 475 - NV_LLDP_NB_TX_CHASSIS_ID_TLV Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.51 NV_LLDP_NB_TX_PORT_ID_TLV

Table 476 - NV_LLDP_NB_TX_PORT_ID_TLV Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 477 - NV_LLDP_NB_TX_PORT_ID_TLV Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.52 NV_LLDP_NB_TX_TTL_TLV

Table 478 - NV_LLDP_NB_TX_TTL_TLV Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																::

Table 479 - NV_LLDP_NB_TX_TTL_TLV Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.53 NV_LLDP_NB_TX_OPTIONAL_TLV0

Table 480 - NV_LLDP_NB_TX_OPTIONAL_TLV0 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 481 - NV_LLDP_NB_TX_OPTIONAL_TLV0 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.54 NV_LLDP_NB_TX_OPTIONAL_TLV1

Table 482 - NV_LLDP_NB_TX_OPTIONAL_TLV1 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																:

Table 483 - NV_LLDP_NB_TX_OPTIONAL_TLV1 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.55 NV_LLDP_NB_TX_OPTIONAL_TLV2

Table 484 - NV_LLDP_NB_TX_OPTIONAL_TLV2 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 485 - NV_LLDP_NB_TX_OPTIONAL_TLV2 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.56 NV_LLDP_NB_TX_OPTIONAL_TLV3

Table 486 - NV_LLDP_NB_TX_OPTIONAL_TLV3 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																::

Table 487 - NV_LLDP_NB_TX_OPTIONAL_TLV3 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.57 NV_LLDP_NB_TX_OPTIONAL_TLV4

Table 488 - NV_LLDP_NB_TX_OPTIONAL_TLV4 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 489 - NV_LLDP_NB_TX_OPTIONAL_TLV4 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.58 NV_LLDP_NB_TX_OPTIONAL_TLV5

Table 490 - NV_LLDP_NB_TX_OPTIONAL_TLV5 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 491 - NV_LLDP_NB_TX_OPTIONAL_TLV5 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.59 NV_LLDP_NB_TX_OPTIONAL_TLV6

Table 492 - NV_LLDP_NB_TX_OPTIONAL_TLV6 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																...

Table 493 - NV_LLDP_NB_TX_OPTIONAL_TLV6 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.60 NV_LLDP_NB_TX_OPTIONAL_TLV7

Table 494 - NV_LLDP_NB_TX_OPTIONAL_TLV7 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_tlv[0]								lldp_tlv[1]								lldp_tlv[2]								lldp_tlv[3]								00h
lldp_tlv[4]								lldp_tlv[5]								lldp_tlv[6]								lldp_tlv[7]								04h
...																																::

Table 495 - NV_LLDP_NB_TX_OPTIONAL_TLV7 Field Descriptions

Offset	Bits	Name	Type	Description
00h- ...	31:24	lldp_tlv[...]	BYTES	This parameter holds a raw ChassisID LLDP TLV to be sent by the client. When undefined, the firmware will auto generate this mandatory TLV.

11.7.61 NV_KEEP_LINK_UP

Table 496 - NV_KEEP_LINK_UP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																										auto power save link down	do not clear port stats	keep link up on standby	keep link up on boot	keep_ib link up	keep eth link up	00h

Table 497 - NV_KEEP_LINK_UP Field Descriptions

Offset	Bits	Name	Type	Description
00h	5	auto_power_save_link_down	BOOL	When set to TRUE, the port will automatically power down when it senses the link is disconnected. mlxconfig_name: AUTO_POWER_SAVE_LINK_DOWN supported_from_version: 3
	4	do_not_clear_port_stats	BOOL	When set, the port statistic counters are not cleared on first host init. When cleared the port statistics are cleared on first host init mlxconfig_name: DO_NOT_CLEAR_PORT_STATS supported_from_version: 2
	3	keep_link_up_on_standby	BOOL	When set, the NIC keeps the link up from power-up until the server is turned on, and when the server is in standby mode (PCI L2/3 state). Note: The link will be kept up only if the server can provide enough power during standby (via PCI rails or AUX power). mlxconfig_name: KEEP_LINK_UP_ON_STANDBY supported_from_version: 1
	2	keep_link_up_on_boot	BOOL	When set, the NIC keeps the link up as long as the server is not in standby mode and a driver is not initialized. mlxconfig_name: KEEP_LINK_UP_ON_BOOT supported_from_version: 1
	1	keep_ib_link_up	BOOL	When set, the NIC keeps the link up as long as the server is not in standby mode (InfiniBand only). mlxconfig_name: KEEP_IB_LINK_UP supported_from_version: 1
	0	keep_eth_link_up	BOOL	When set, the NIC keeps the link up as long as the server is not in standby mode (Ethernet only). mlxconfig_name: KEEP_ETH_LINK_UP

11.7.62 NV_QOS_CONF

Table 498 - NV_QOS_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 499 - NV_QOS_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	11:8	num_of_pfc	UNSIGNED	Number of IEEE priorities that may simultaneously support flow control. (See pfc_cap_admin DCBX_PARAM Register) mlxconfig_name: NUM_OF_PFC supported_from_version: 1
	7:4	num_of_tc	ENUM	Number of traffic classes, when DCB-X is enabled, this is the maximum number of TC that can negotiate with the remote peer. (see num_of_tc_admin in DCBX_PARAM Register) 0x0: _8_TCs 0x1: _1_TC 0x2: _2_TCs 0x3: _3_TCs 0x4: _4_TCs 0x5: _5_TCs 0x6: _6_TCs 0x7: _7_TCs mlxconfig_name: NUM_OF_TC
	3:0	num_of_vl	ENUM	Number of Infiniband Virtual Lanes for this port, (see vl_admin in PVLC Register) 0x1: _1_VL - VL0 only 0x2: _2_VLs - VL0, VL1 0x3: _4_VLs - VL0..VL3 0x4: _8_VLs - VL0..VL7 0x5: _15_VLs --VL0, VL14 (VL15 is always present for MADs) mlxconfig_name: NUM_OF_VL

11.7.63NV_QOS_CAP

Table 500 - NV_QOS_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				max- _num_pfc				max- _num_of_tc				max- _num_of_vl				00h

Table 501 - NV_QOS_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	11:8	max_num_pfc	UNSIGNED	Maximum number of priority flow control supported_from_version: 1
	7:4	max- _num_of_tc	UNSIGNED	Maximum number of traffic classes supported by the NIC, where 0 => 8 TC
	3:0	max- _num_of_vl	ENUM	Maximum number of VLs supported by the HBA. 0x1: _1_VL - VL0 only 0x2: _2_VLs - VL0, VL1 0x3: _4_VLs - VL0..VL3 0x4: _8_VLs - VL0..VL7 0x5: _15_VLs -- VL0..VL14 (VL15 is always present for MADs)

11.7.64 NV_ROM_IB_DHCP_CONF

Table 502 - NV_ROM_IB_DHCP_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				client_iden- tifier																											00h	

Table 503 - NV_ROM_IB_DHCP_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	27:24	client_identi- fier	ENUM	Select if the expansion ROM should add a Client Identifier to DHCP DISCOVER and REQUEST packets over InfiniBand. 0x0: ROM_DEFAULT 0x1: NO_CLIENT_IDENTIFIER 0x2: SEND_CLIENT_IDENTIFIER When set, the UEFI driver modifies the DHCP packet sent by the upper layers.

11.7.65 NV_SECURE_HOST

Table 504 - NV_SECURE_HOST Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
state																																00h
																																04h
key[0]								key[1]								key[2]								key[3]								08h
key[4]								key[5]								key[6]								key[7]								0Ch
																																10h-1Ch

Table 505 - NV_SECURE_HOST Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:24	state	ENUM	0x0: UNLOCKED 0x1: LOCKED
08h-0Ch	31:24	key[8]	BYTES	

11.7.66 NV_KDNET_DATA

Table 506 - NV_KDNET_DATA Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
kdnet_en																															00h	

Table 507 - NV_KDNET_DATA Field Descriptions

Offset	Bits	Name	Type	DESCRIPTION
00h	31	kdnet_en	BOOL	KDNET PF is enabled when this bit is set.

11.7.67 NV_PCI_SWITCH_EVB_CONF

Table 508 - NV_PCI_SWITCH_EVB_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pci_switch_lanes_remapping_en																																00h
																																04h
																																08h

Table 509 - NV_PCI_SWITCH_EVB_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	pci_switch_lanes_remapping_en	BOOL	When set allows remapping of the PCI switch lanes to slots according to the PCI_SW_LANES_MAPPING parameter. Note: This TLV may only be available in ConnectX-5 (and above) evaluation boards and in other boards during the bringup phase. mlxconfig_name: PCI_SW_LANES_REMAPPING_EN

Table 509 - NV_PCI_SWITCH_EVB_CONF Field Descriptions

Offset	Bits	Name	Type	Description
04h-08h	31:24	pci_switch_lanes_mapping[8]	BINARY	Evaluated from the LSB, each bit represents the mapping of two PCI lanes to independent PCI buses where a value of zero indicates that the corresponding lanes are the first lanes of a new bus, and a value of 1 indicates that the corresponding lanes belongs to the same bus as the previous lanes. [Internal] For ConnectX-5 bit 0 always equals to zero as it represents that lanes 0 and 1 are always connected to the host. Examples: 11111110b => all 16 lanes are mapped to the host. The actual bus width is learned dynamically so it may also be 8 or 4 lanes. 00000000b => The host bus width is up to 2 lanes and there are additional 7 independent buses. 11101110b => The host bus is up to 8 lanes and there is an addition bus of up to 8 lanes. mlxconfig_name: PCI_SW_LANES_MAPPING

11.7.68 NV_PCI_SWITCH_EVB_CAP

Table 510 - NV_PCI_SWITCH_EVB_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								number_of_pci_switch_lanes				00h				

Table 511 - NV_PCI_SWITCH_EVB_CAP Field Descriptions

Offset	Bits	Name	Type	DESCRIPTION
00h	7:0	number_of_pci_switch_lanes	UNSIGNED	The number of lanes that can be configured in the PCI switch using the PCI_SW_LANES_MAPPING parameter.

11.7.69NV_FPGA_DATA

Table 512 - NV_FPGA_DATA Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
image_bank																																00h

Table 513 - NV_FPGA_DATA Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:30	image_bank	UNSIGNED	Select which FPGA Image to load. 0x0: FACTORY_IMAGE - Factory-default Image 0x1: USER_IMAGE_1 - User configured Image #1 Other - Reserved

11.7.70NV_EXTERNAL_PORT_CTRL

Table 514 - NV_EXTERNAL_PORT_CTRL Layout

Offset	00h	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	port owner	allow_rd_counters	renew_on_change	tracer_enable																													

Table 515 - NV_EXTERNAL_PORT_CTRL Field Descriptions

Offset	Bits	Name	Type	Description
00h	3	tracer_enable	BOOL	If set, indicates the function may own the Device Tracer. mlxconfig_name: TRACER_ENABLE
	2	reneg_on_change	BOOL	When cleared, a link-down/up sequence from the driver (PAOS) triggers re-negotiation of the link speed and parameters with the remote peer. When set, a link down/up sequence will trigger re-negotiation only if any link parameters were changed by a driver since last link negotiation. This parameter is applicable when the port_owner parameter is set for this PF mlxconfig_name: RENEG_ON_CHANGE supported_from_version:1
	1	allow_rd_counters	BOOL	If set, indicates the host function which is allowed to read counters of external physical port. mlxconfig_name: ALLOW_RD_COUNTERS
	0	port_owner	BOOL	If set, indicates the function of the host's own the external physical port. mlxconfig_name: PORT_OWNER

11.7.71 NV_ROCE_1_5_CONF

Table 516 - NV_ROCE_1_5_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								roce_over_ip_next_protocol				00h				

Table 517 - NV_ROCE_1_5_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	7:0	roce_over_ip_next_protocol	UNSIGNED	The next protocol value set in the IPv4/IPv6 packets for RoCE v1.5. [Internal] The default is 0xFE. mlxconfig_name: ROCE_NEXT_PROTOCOL

11.7.72 NV_HOST_TO_BMC

Table 518 - NV_HOST_TO_BMC Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																host2bmc	00h

Table 519 - NV_HOST_TO_BMC Field Descriptions

Offset	Bits	Name	Type	Description
00h	0	host2bmc	UNSIGNED	When set, the host can communicate directly with the BMC.

11.7.73 NV_INTERNAL_RESOURCE_HAIRPIN_CONF

Table 520 - NV_INTERNAL_HAIRPIN_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
log_hpin_num_descriptor[0]								log_hpin_num_descriptor[1]								log_hpin_num_descriptor[2]								log_hpin_num_descriptor[3]								00h
log_hpin_num_descriptor[4]								log_hpin_num_descriptor[5]								log_hpin_num_descriptor[6]								log_hpin_num_descriptor[7]								04h
log_hpin_data_size[0]								log_hpin_data_size[1]								log_hpin_data_size[2]								log_hpin_data_size[3]								08h
log_hpin_data_size[4]								log_hpin_data_size[5]								log_hpin_data_size[6]								log_hpin_data_size[7]								0Ch

Table 521 - NV_INTERNAL_HAIRPIN_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	log_hpin_num_descriptor[8]	UNSIGNED	Log(base 2) of the number of packets descriptors allocated internally for hairpin for a given IEEE802.1p priority i. 0 means that no descriptors are allocated for this priority and traffic with this priority will be dropped. mlxconfig_name: ESWITCH_HAIRPIN_DESCRIPTOR

Table 521 - NV_INTERNAL_HAIRPIN_CONF Field Descriptions

Offset	Bits	Name	Type	Description
08h-0Ch	31:24	log_hpin_data_size[8]	UNSIGNED	Log(base 2) of the buffer size (in bytes) allocated internally for hairpin for a given IEEE802.1p priority i. 0 means no buffer for this priority and traffic with this priority will be dropped. mlxconfig_name: ESWITCH_HAIRPIN_TOT_BUFFER_SIZE

11.7.74 NV_INTERNAL_HAIRPIN_CAP

Table 522 - NV_INTERNAL_HAIRPIN_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
log_max_hpin_total_num_descriptors								log_max_hpin_total_data_size								log_max_hpin_num_descriptor_per_prio								log_max_hpin_data_size_per_prio								00h

Table 523 - NV_INTERNAL_HAIRPIN_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:24	log_max_hpin_total_num_descriptors	UNSIGNED	Log2 of the maximal total number of packet descriptors allocated for eswitch hairpin (sum of all priorities).
	23:16	log_max_hpin_total_data_size	UNSIGNED	Log(base 2) of the maximal total data buffer size allocated for eswitch hairpin (sum of all priorities)
	15:8	log_max_hpin_num_descriptor_per_prio	UNSIGNED	Log(base 2) of the maximal number of packet descriptors allocated for eswitch hairpin per priority
	7:0	log_max_hpin_data_size_per_prio	UNSIGNED	Log(base 2) of the maximal data buffer size allocated for eswitch hairpin per priority

11.7.75NV_LLDP_NB_CONF

Table 524 - NV_LLDP_NB_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_nb_tx_mode		lldp_nb_rx_mode		lldp_nb_dcbx																lldp_msg_tx_interval											00h	
																																04h
lldp_nb_tx_buffer_size															lldp_nb_rx_buffer_size																	08h

Table 525 - NV_LLDP_NB_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:30	lldp_nb_tx_mode	ENUM	Select which LLDP TLV will be generated by the NIC 0x0: OFF - NIC internal LLDP client will not send LLDP frames 0x1: MANDATORY - Transmits only mandatory LLDP BPDU (ChassisID, PortID & TTL) 0x2: ALL - Transmits optional LLDP BPDU if configured mlxconfig_name: LLDP_NB_TX_MODE
	29:28	lldp_nb_rx_mode	ENUM	Enable the internal LLDP client, and define which TLV it will process. 0x0: OFF - Doesnot listen to incoming LLDP BPDU (incoming LLDP frames will be routed to the host) 0x1: MANDATORY - Listen to incoming LLDP frames, store only the mandatory LLDP BPDUs (1..3) 0x2: ALL - Receive and store all incoming LLDP BPDUs mlxconfig_name: LLDP_NB_RX_MODE
	27	lldp_nb_dcbx	BOOL	Enables DCBX (applicable when LLDP_NB_TX_MODE and LLDP_NB_RX_MODE are in ALL mode). mlxconfig_name: LLDP_NB_DCBX supported_from_version:1
	11:0	lldp_msg_tx_interval	UNSIGNED	Interval for sending LLDP NB PDUs in seconds

Table 525 - NV_LLDP_NB_CONF Field Descriptions

Offset	Bits	Name	Type	Description
08h	31:16	lldp_nb_tx- buffer_size	UNSIGNED	The size in bytes of LLDP client must reserve for transmitted TLVs. The buffer size must be big enough to hold all mandatory TLVs (chassisID, PortID & TTL) as well as all optional TLVs that the client needs to send.
	15:0	lldp_nb_rx- buffer_size	UNSIGNED	The size in bytes LLDP client must reserve for incoming TLVs

11.7.76 NV_LLDP_NB_CAP

Table 526 - NV_LLDP_NB_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lldp_nb_tx_cap			lldp_nb_rx_cap		lldp_nb_deb_x_en																						00h					
max_lldp_nb_tx_buffer_size															max_lldp_nb_rx_buffer_size													04h				
																								max_num_of_tx_optional_lldp_tlv				08h				

Table 527 - NV_LLDP_NB_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:3 0	lldp_nb_tx_cap	ENUM	0x0: NA - LLDP transmission not supported 0x1: MANDATORY - The LLDP client only support sending the mandatory LLDP NB TLVs (ChassisID, PortID & TTL) 0x2: ALL - The LLDP client support sending mandatory and optional TLVs 0x3: reserved
	29:2 8	lldp_nb_rx_cap	ENUM	0x0: NA - Reception and storing of LLDP frames is not supported (incoming frames will be forwarded to the host) 0x1: MANDATORY - The client can only store received mandatory LLDP-TLVs 0x2: ALL - The client can receive and store incoming LLDP TLVs
	27	lldp_nb_dcbx_en	BOOL	DCB-X supported supported_from_version:1
04h	31:1 6	max_lldp_nb_tx_buffer_size	UNSIGNED	The maximum supported buffer size for outgoing LLDP TLVs
	15:0	max_lldp_nb_rx_buffer_size	UNSIGNED	The maximum supported buffer size for incoming LLDP TLVs
08h	6:0	max_num_of_tx_optional_lldp_tlv	UNSIGNED	The maximum number of optional LLDP TLVs that are supported by the firmware. The firmware is searching for LLDP NB TLV to send from type 0x110 to type (0x110 + this field) Default: 16

11.7.77 NV_ROCE_CC

Table 528 - NV_ROCE_CC Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
roce_cc_algorithm																								roce_cc_enable_pri				00h				

Table 529 - NV_ROCE_CC Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	roce_cc_algorithm	ENUM	Select RDMA over Converged Ethernet (RoCE) algorithm 0x0: ECN 0x1: QCN mlxconfig_name: ROCE_CC_ALGORITHM
	7:0	roce_cc_enable_pri	UNSIGNED	Each bit in this mask indicates if the RoCE should be enabled on the n-th IEEE priority. mlxconfig_name: ROCE_CC_PRIO_MASK

11.7.78 NV_ROCE_CC_ECN

Table 530 - NV_ROCE_CC_ECN Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
clamp_tgt_rate																																00h
																rpg_time_reset												04h				
																rpg_byte_reset												08h				
																												rpg_threshold	0Ch			
rpg_max_rate																												10h				

Table 530 - NV_ROCE_CC_ECN Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
															rpg_ai_rate															14h		
															rpg_hai_rate															18h		
																												rpg_gd	1Ch			
																							rpg_min_dec_fac					20h				
															rpg_min_rate													24h				
															rate_to_set_on_first_cnp													28h				
																					dce_tcp_g							2Ch				
															dce_tcp_rtt													30h				
															rate_reduce_monitor_period													34h				
																					initial_alpha_value							38h				
															min_time_between_cnps													3Ch				
																		cnp_dscp								cnp_prio_mode	cnp_802p_prio		40h			

Table 531 - NV_ROCE_CC_ECN Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	clamp_tgt_rate	BOOL	If set, whenever a CNP is processed, the target rate is updated to be the current rate. mlxconfig_name: CLAMP_TGT_RATE
	30	clamp_tgt_rate_after_time_inc	BOOL	Default: True When receiving a CNP, the target rate should be updated if the transmission rate was increased due to the timer, and not only due to the byte counter mlxconfig_name: CLAMP_TGT_RATE_AFTER_TIME_INC
04h	16:0	rpg_time_reset	UNSIGNED	Time between rate increases if no CNPs are received. Given in microseconds. Default: 2 mlxconfig_name: RPG_TIME_RESET
08h	14:0	rpg_byte_reset	UNSIGNED	Transmitted data between rate increases if no CNPs are received. Given in Bytes (0=DISABLED) Default: 150 mlxconfig_name: RPG_BYTE_RESET

Table 531 - NV_ROCE_CC_ECN Field Descriptions

Offset	Bits	Name	Type	Description
0Ch	4:0	rpg_threshold	UNSIGNED	The number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states. Default: 5 mlxconfig_name: RPG_THRESHOLD
10h	31:0	rpg_max_rate	UNSIGNED	The maximum rate, in Mbits per second, at which an RP can transmit. Once this limit is reached, the RP rate limited is released and the flow is not rate limited any more (0=Full Port Speed). Default: 0 (Full port speed) mlxconfig_name: RPG_MAX_RATE
14h	16:0	rpg_ai_rate	UNSIGNED	The rate, in megabits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE. Default: 10 mlxconfig_name: RPG_AI_RATE
18h	16:0	rpg_hai_rate	UNSIGNED	The rate, in megabits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state. Default: 50 mlxconfig_name: RPG_HAI_RATE
1Ch	3:0	rpg_gd	UNSIGNED	If a CNP is received, the flow rate is reduced at the beginning of the next rate_reduce_monitor_period interval to $(1 - \text{Alpha}/\text{Gd}) * \text{CurrentRate}$. rpg_gd is given as $\log_2(\text{Gd})$, where Gd may only be powers of 2. Default: 7 mlxconfig_name: RPG_GD
20h	7:0	rpg_min_dec_fac	UNSIGNED	The minimum factor by which the current transmit rate can be changed when processing a CNP. Value is given as a percentage (1-100). Default: 50 mlxconfig_name: RPG_MIN_DEC_FAC
24h	16:0	rpg_min_rate	UNSIGNED	The minimum value, in megabits per second, for rate to limit. Default: 2000 mlxconfig_name: RPG_MIN_RATE
28h	16:0	rate_to_set_on_first_cnp	UNSIGNED	The rate that is set for the flow when a rate limiter is allocated to it upon first CNP received, in Mbps (=Full Port Speed). Default: 0 Port speed mlxconfig_name: RATE_TO_SET_ON_FIRST_CNP

Table 531 - NV_ROCE_CC_ECN Field Descriptions

Offset	Bits	Name	Type	Description
2Ch	9:0	dce_tcp_g	UNSIGNED	Used to update the congestion estimator (alpha) once every dce_tcp_rtt microseconds, according to the equation: $\text{Alpha} = (\text{cnp\_received} * \text{dceTcpG}) + (1 - \text{dceTcpG}) * \text{alpha}$. dceTcpG is divided by 2^{10} . cnp_received is set to one if a CNP was received for this flow during period since the previous update and the current update Default: 64 mlxconfig_name: DCE_TCP_G
30h	16:0	dce_tcp_rtt	UNSIGNED	The time between updates of the alpha value, in microseconds. Default: 2 mlxconfig_name: DCE_TCP_RTT
34h	16:0	rate_reduce_monitor_period	UNSIGNED	The minimum time between 2 consecutive rate reductions for a single flow. Rate reduction will occur only if a CNP is received during the relevant time interval. Default: 2 mlxconfig_name: RATE_REDUCE_MONITOR_PERIOD
38h	9:0	initial_alpha_value	UNSIGNED	The initial value of alpha to use when receiving the first CNP for a flow. Expressed in a fixed point fraction of 2^{10} . mlxconfig_name: INITIAL_ALPHA_VALUE
3Ch	16:0	min_time_between_cnps	UNSIGNED	Minimum time between sending CNPs from the port, in microseconds. Default: 0 mlxconfig_name: MIN_TIME_BETWEEN_CNPS
40h	13:8	cnp_dscp	UNSIGNED	The DiffServ Code Point of the generated CNP for this port. mlxconfig_name: CNP_DSCP
	3	cnp_prio_mode	BOOL	If TRUE, CNP packets for this port contain priority from a received request. If FALSE, CNP responses use value set by CNP_802P_PRIO. mlxconfig_name: CNP_RES_PRIO_MODE supported_from_version: 1
	2:0	cnp_802p_prio	UNSIGNED	The 802.1p priority value of the generated CNP for this port Default: 7 mlxconfig_name: CNP_802P_PRIO

11.7.79 NV_ROCE_CC_QCN

Table 532 - NV_ROCE_CC_QCN Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
															rpg_time_reset													04h				
																	rpg_byte_reset											08h				
																									rpg_threshold			0Ch				
rpg_max_rate																															10h	
															rpg_ai_rate													14h				
															rpg_hai_rate													18h				
																									rpg_gd			1Ch				
																							rpg_min_dec_fac						20h			
															rpg_min_rate													24h				

Table 533 - NV_ROCE_CC_QCN Field Descriptions

Offset	Bits	Name	Type	Description
04h	16:0	rpg_time_reset	UNSIGNED	Time between rate increases if no CNPs are received. Given in u-seconds. Default: 15000
08h	14:0	rpg_byte_reset	UNSIGNED	Transmitted data between rate increases if no CNPs are received. Given in Bytes (0 = DISABLED). Default: 150000
0Ch	4:0	rpg_threshold	UNSIGNED	The number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states. Default: 5
10h	31:0	rpg_max_rate	UNSIGNED	The maximum rate, in Mbits per second, at which an RP can transmit. Once this limit is reached, the RP rate limited is released and the flow stops being limited (0=Full Port Speed) Default: 0 (Full port speed)
14h	16:0	rpg_ai_rate	UNSIGNED	The rate, in Mbits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE. Default: 5

Table 533 - NV_ROCE_CC_QCN Field Descriptions

Offset	Bits	Name	Type	Description
18h	16:0	rpg_hai_rate	UNSIGNED	The rate, in Mbits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state Default: 50
1Ch	3:0	rpg_gd	UNSIGNED	When a CNP is received, the flow rate is reduced at the beginning of the next rate_reduce_monitor_period interval to $(1 - \text{Alpha}/\text{Gd}) * \text{CurrentRate}$. rpg_gd is given as $\log_2(\text{Gd})$, where Gd may only be powers of 2. Default: 7
20h	7:0	rpg_min_dec_fac	UNSIGNED	The minimum factor by which the current transmit rate can be changed when processing a CNP. Value is given as a percentage (1-100). Default: 50
24h	16:0	rpg_min_rate	UNSIGNED	The minimum value, in Mbits per second, for rate to limit. Default: 10

11.7.80 FILE_APPLICABLE_TO

Table 534 - FILE_APPLICABLE_TO Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
psid[0]								psid[1]								psid[2]								psid[3]								00h
psid[4]								psid[5]								psid[6]								psid[7]								04h
...																																...
psid[12]								psid[13]								psid[14]								psid[15]								0Ch
psid_branch[0]								psid_branch[1]								psid_branch[2]								psid_branch[3]								10h
psid_branch[4]								psid_branch[5]								psid_branch[6]								psid_branch[7]								14h
...																																...
psid_branch[12]								psid_branch[13]								psid_branch[14]								psid_branch[15]								1Ch

Table 535 - FILE_APPLICABLE_TO Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	psid[16]	STRING	The PSID of the NIC that this file is applicable for. This ASCII string must be zero terminated if shorter than 16 characters.

Table 535 - FILE_APPLICABLE_TO Field Descriptions

Offset	Bits	Name	Type	Description
10h-1Ch	31:24	psid_branch[16]	STRING	The PSID-BRANCH that this file is applicable to. This ASCII string must be zero terminated if shorter than 16 characters. If PSID-Branch is not defined, the string is empty.

11.7.81 FILE_COMMENT

Table 536 - FILE_COMMENT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
text[0]								text[1]								text[2]								text[3]								00h
text[4]								text[5]								text[6]								text[7]								04h
...																																...

Table 537 - FILE_COMMENT Field Descriptions

Offset	Bits	Name	Type	Description
00h-...	31:24	text[...]	STRING	The PSID of the NIC that this file is applicable for. This ASCII string must be zero terminated if shorter than 16 characters.

11.7.82 FILE_SIGNATURE

Table 538 - FILE_SIGNATURE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
signature_uuid[0]								signature_uuid[1]								signature_uuid[2]								signature_uuid[3]								00h
...																																...
signature_uuid[12]								signature_uuid[13]								signature_uuid[14]								signature_uuid[15]								0Ch
keypair_uuid[0]								keypair_uuid[1]								keypair_uuid[2]								keypair_uuid[3]								10h
...																																...
keypair_uuid[12]								keypair_uuid[13]								keypair_uuid[14]								keypair_uuid[15]								1Ch
signature[0]								signature[1]								signature[2]								signature[3]								20h

Table 538 - FILE_SIGNATURE Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
...																																...
signature[252]								signature[253]								signature[254]								signature[255]								11Ch
																																120h-13Ch

Table 539 - FILE_SIGNATURE Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	signature_uuid[16]	BYTES	Time based UUID for this signature.
10h-1Ch	31:24	key-pair_uuid[16]	BYTES	The UUID of the key-pair used for signing this file. An all zero UUID means that the signature is a SHA256 of the image.
20h-11Ch	31:24	signature[256]	BYTES	The signature itself

11.7.83 FILE_DBG_FW_TOKEN_ID

Table 540 - FILE_DBG_FW_TOKEN_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
running_version																																00h
debug_version																																04h

Table 541 - FILE_DBG_FW_TOKEN_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:0	running_version	BINARY	The current running version.
04h	31:0	debug_version	BINARY	The debug firmware version approved by this token.

11.7.84 FILE_CS_TOKEN_ID

Table 542 - FILE_CS_TOKEN_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
running_version																															00h	

Table 543 - FILE_CS_TOKEN_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:0	running_version	BINARY	The current running version.

11.7.85 FILE_PUBLIC_KEY

This TLV is used for creating PUBLIC_KEYS section binary for secure firmware.

Table 544 - FILE_PUBLIC_KEY Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
fw_en	cs_token_en	vendor_nvconf_en	mlnx_nvconf_en	fw_en																			auth_type										00h
																																04h	
																																08h	
public_key_exp																																0Ch	
keypair_uuid[0]								keypair_uuid[1]								keypair_uuid[2]								keypair_uuid[3]								10h	
...																																...	
keypair_uuid[12]								keypair_uuid[13]								keypair_uuid[14]								keypair_uuid[15]								1Ch	
key[0]								key[1]								key[2]								key[3]								20h	
...																																...	
key[252]								key[253]								key[254]								key[255]								11Ch	

Table 545 - FILE_PUBLIC_KEY Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	fw_en	BOOL	This key can be used for authenticating firmware, DBG_FW and DBG tokens
	30	cs_token_en	BOOL	This key can be used for authenticating CS tokens at OEM level.
	29	vendor_nv-conf_en	BOOL	This key can be used for authenticating NVCONFIG files at OEM level.
	28	mlnx_nvconf_en	BOOL	This key can be used for authenticating NVCONFIG files at MLNX level.
	27	frc_en	BOOL	This key can be used for authenticating Factory Re Configuration responses.
	7:0	auth_type	ENUM	The authentication mechanism used with this key 0x0: INVALID_ENTRY 0x1: SHA256_DIGEST 0x3: _2048BIT_RSASSA_PKCS1_v1_5_WITH_SHA256
0Ch	31:0	public_key_exp	UNSIGNED	The public key exponent. In most cases, the standard exponent (65537) will be used (per RFC4871).
10h-1Ch	31:24	key-pair_uuid[16]	BYTES	UUID of this key. The UUID is created by the sign server when it generates a new RSA key-pair.
20h-11Ch	31:24	key[256]	BYTES	A 2048 bit public-key (0x100 bytes)

11.7.86FILE_MAC_ADDR_LIST

Table 546 - FILE_MAC_ADDR_LIST Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
mac_address[0]								mac_address[1]								mac_address[2]								mac_address[3]								00h
...																																...
mac_address[252]								mac_address[253]								mac_address[254]								mac_address[255]								FCh

Table 547 - FILE_MAC_ADDR_LIST Field Descriptions

Offset	Bits	Name	Type	Description
00h-FCh	31:24	mac_address[256]	BYTES	MAC Address list

11.7.87 FILE_SIGNATURE_4096_A

This TLV is used with “[FILE_SIGNATURE_4096_B](#)” for creating 4096 bit file signature. All signature parameters of these two TLVs must match to indicate a single signature.

Table 548 - FILE_SIGNATURE_4096_A Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
signature_uuid[0]								signature_uuid[1]								signature_uuid[2]								signature_uuid[3]								00h
...																																...
signature_uuid[12]								signature_uuid[13]								signature_uuid[14]								signature_uuid[15]								0Ch
keypair_uuid[0]								keypair_uuid[1]								keypair_uuid[2]								keypair_uuid[3]								10h
...																																...
keypair_uuid[12]								keypair_uuid[13]								keypair_uuid[14]								keypair_uuid[15]								1Ch
signature[0]								signature[1]								signature[2]								signature[3]								20h
...																																...
signature[252]								signature[253]								signature[254]								signature[255]								11Ch

Table 549 - FILE_SIGNATURE_4096_A Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	signature_uuid[16]	BYTES	Time based UUID for this signature.
10h-1Ch	31:24	keypair_uuid[16]	BYTES	The UUID of the key-pair used for signing this file. An all zero UUID means that the signature is a SHA256 of the image.
20h-11Ch	31:24	signature[256]	BYTES	First half of the signature itself

11.7.88 FILE_SIGNATURE_4096_B

This TLV is used with “[FILE_SIGNATURE_4096_A](#)” for creating 4096 bit file signature. All signature parameters of these two TLVs must match to indicate a single signature.

Table 550 - FILE_SIGNATURE_4096_B Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
signature_uuid[0]								signature_uuid[1]								signature_uuid[2]								signature_uuid[3]								00h
...																																...
signature_uuid[12]								signature_uuid[13]								signature_uuid[14]								signature_uuid[15]								0Ch
keypair_uuid[0]								keypair_uuid[1]								keypair_uuid[2]								keypair_uuid[3]								10h
...																																...
keypair_uuid[12]								keypair_uuid[13]								keypair_uuid[14]								keypair_uuid[15]								1Ch
signature[0]								signature[1]								signature[2]								signature[3]								20h
...																																...
signature[252]								signature[253]								signature[254]								signature[255]								11Ch

Table 551 - FILE_SIGNATURE_4096_B Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	signature_uuid[16]	BYTES	Time based UUID for this signature.
10h-1Ch	31:24	keypair_uuid[16]	BYTES	The UUID of the key-pair used for signing this file. An all zero UUID means that the signature is a SHA256 of the image.
20h-11Ch	31:24	signature[256]	BYTES	Second half of the signature itself

11.7.89 FILE_PUBLIC_KEY_4096

This TLV is not accessible through tools as it is larger than MAX TLV. it is in PRM for automation purposes only.

Table 552 - FILE_PUBLIC_KEY_4096 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
fw_en	cs_token_en	vendor_nvconf_en	mlnx_nvconf_en	fw_en																			auth_type					00h								
																																				04h
																																				08h
					public_key_exp																															0Ch
keypair_uuid[0]					keypair_uuid[1]					keypair_uuid[2]					keypair_uuid[3]					10h																
...																															...					
keypair_uuid[12]					keypair_uuid[13]					keypair_uuid[14]					keypair_uuid[15]					1Ch																
key[0]					key[1]					key[2]					key[3]					20h																
...																															...					
key[508]					key[509]					key[510]					key[511]					21Ch																

Table 553 - FILE_PUBLIC_KEY_4096 Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	fw_en	BOOL	This key can be used for authenticating firmware, DBG_FW and DBG tokens
	30	cs_token_en	BOOL	This key can be used for authenticating CS tokens at OEM level.
	29	vendor_nv-conf_en	BOOL	This key can be used for authenticating NVCONFIG files at OEM level.
	28	mlnx_nvconf_en	BOOL	This key can be used for authenticating NVCONFIG files at MLLNX level.
	27	frc_en	BOOL	This key can be used for authenticating Factory Re Configuration responses.
	7:0	auth_type	ENUM	The authentication mechanism used with this key 0x0: INVALID_ENTRY 0x1: SHA256_DIGEST - not supported for this Key. 0x3: _2048BIT_RSASSA_PKCS1_v1_5_WITH_SHA256 - not supported for this Key. 0x4: _4096BIT_RSASSA_PKCS1_v1_5_WITH_SHA512
0Ch	31:0	public_key_exp	UNSIGNED	The public key exponent. In most cases, the standard exponent (65537) will be used (per RFC4871).
10h-1Ch	31:24	key-pair_uuid[16]	BYTES	UUID of this key. The UUID is created by the sign server when it generates a new RSA key-pair.
20h-21Ch	31:24	key[512]	BYTES	4096 bit public-key

11.7.90FILE_PUBLIC_KEY_4096_A

[Internal][Unimplemented]

This TLV is used with “FILE_PUBLIC_KEY_4096_B” for creating PUBLIC_KEYS of 4096 section binary for secure firmware. All key parameters of these two TLVs must match.

Table 554 - FILE_PUBLIC_KEY_4096_A Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fw_en	cs_token_en	vendor_nvconf_en	mlnx_nvconf_en	fw_en																			auth_type					00h				
																																04h
																																08h
					public_key_exp																											0Ch
keypair_uuid[0]					keypair_uuid[1]					keypair_uuid[2]					keypair_uuid[3]					10h												
...																											...					
keypair_uuid[12]					keypair_uuid[13]					keypair_uuid[14]					keypair_uuid[15]					1Ch												
key[0]					key[1]					key[2]					key[3]					20h												
...																											...					
key[252]					key[253]					key[254]					key[255]					11Ch												

Table 555 - FILE_PUBLIC_KEY_4096_A Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	fw_en	BOOL	This key can be used for authenticating firmware, DBG_FW and DBG tokens
	30	cs_token_en	BOOL	This key can be used for authenticating CS tokens at OEM level.
	29	vendor_nv-conf_en	BOOL	This key can be used for authenticating NVCONFIG files at OEM level.
	28	mlnx_nvconf_en	BOOL	This key can be used for authenticating NVCONFIG files at MLLNX level.
	27	frc_en	BOOL	This key can be used for authenticating Factory Re Configuration responses.
	7:0	auth_type	ENUM	The authentication mechanism used with this key 0x0: INVALID_ENTRY 0x1: SHA256_DIGEST - not supported for this Key. 0x3: _2048BIT_RSASSA_PKCS1_v1_5_WITH_SHA256 - not supported for this Key. 0x4: _4096BIT_RSASSA_PKCS1_v1_5_WITH_SHA512
0Ch	31:0	public_key_exp	UNSIGNED	The public key exponent. In most cases, the standard exponent (65537) will be used (per RFC4871).
10h-1Ch	31:24	key-pair_uuid[16]	BYTES	UUID of this key. The UUID is created by the sign server when it generates a new RSA key-pair.
20h-11Ch	31:24	key[256]	BYTES	First half of a 4096 bit public-key (0x200 bytes)

11.7.91 FILE_PUBLIC_KEY_4096_B

[Internal][Unimplemented]

This TLV is used with “[FILE_PUBLIC_KEY_4096_A](#)” for creating PUBLIC_KEYS of 4096 section binary for secure firmware. All key parameters of these two TLVs must match.

Table 556 - FILE_PUBLIC_KEY_4096_B Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
fw_en	cs_token_en	vendor_nvconf_en	mlnx_nvconf_en	fric_en																			auth_type												00h	
																																				04h
																																				08h
					public_key_exp																															0Ch
keypair_uuid[0]					keypair_uuid[1]					keypair_uuid[2]					keypair_uuid[3]					10h																
...																															...					
keypair_uuid[12]					keypair_uuid[13]					keypair_uuid[14]					keypair_uuid[15]					1Ch																
key[0]					key[1]					key[2]					key[3]					20h																
...																															...					
key[252]					key[253]					key[254]					key[255]					11Ch																

Table 557 - FILE_PUBLIC_KEY_4096_B Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	fw_en	BOOL	This key can be used for authenticating firmware, DBG_FW and DBG tokens
	30	cs_token_en	BOOL	This key can be used for authenticating CS tokens at OEM level.
	29	vendor_nv-conf_en	BOOL	This key can be used for authenticating NVCONFIG files at OEM level.
	28	mlnx_nvconf_en	BOOL	This key can be used for authenticating NVCONFIG files at MLLNX level.
	27	frc_en	BOOL	This key can be used for authenticating Factory Re Configuration responses.
	7:0	auth_type	ENUM	The authentication mechanism used with this key 0x0: INVALID_ENTRY 0x1: SHA256_DIGEST - not supported for this Key. 0x3: _2048BIT_RSASSA_PKCS1_v1_5_WITH_SHA256 - not supported for this Key. 0x4: _4096BIT_RSASSA_PKCS1_v1_5_WITH_SHA512
0Ch	31:0	public_key_exp	UNSIGNED	The public key exponent. In most cases, the standard exponent (65537) will be used (per RFC4871).
10h-1Ch	31:24	key-pair_uuid[16]	BYTES	UUID of this key. The UUID is created by the sign server when it generates a new RSA key-pair.
20h-11Ch	31:24	key[256]	BYTES	Second half of a 4096 bit public-key (0x200 bytes)

11.7.92NV_MPFS_CONF

Table 558 - NV_MPFS_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ib_routing_mode	snov_ib_routing_mode																						unknown uplink mac flood					duplicat- ed_mac_ac- tion_mode				00h

Table 559 - NV_MPFS_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	ib_routing_mode	ENUM	Select the routing mode for Infiniband among the PFs (and hosts, if applicable) 0x0: GID - each PF has a unique GID 0x1: LID - each PF has a unique LID mlxconfig_name: IB_ROUTING_MODE
	30	sriov_ib_routing_mode	ENUM	Select the IB routing mode for Virtual Functions [Internal] This parameter should only be configurable and visible if the port is in IB mode and sriov is enabled. 0x0: GID 0x1: LID mlxconfig_name: SRIOV_IB_ROUTING_MODE
	7	<u>unknown_uplink_mac_flood</u>	<u>BOOL</u>	<u>Defines the forwarding behavior in MPFS for packets arriving from the network (uplink) with destination MAC address that does not appear in the MPFS FDB.</u> <u>When TRUE, these packets are flooded to all local MPFS ports.</u> <u>When FALSE, these packets are dropped.</u> <u>mlxconfig_name: UNKNOWN_UPLINK_MAC_FLOOD</u>
	3:0	duplicate_mac_action_mode	ENUM	Defines the forwarding behavior in MPFS for MACs which are duplicated in more than one PF. 0x0: LAST_CFG - last PF which added the MAC will receive the traffic. 0x1: LOAD_BALANCE - MPFS will load-balance ipv4 traffic for packets with destination MAC addresses that belong to more than one PF between all PFs that have this MAC mlxconfig_name: DUP_MAC_ACTION

11.7.93NV_MPFS_CAP

Table 560 - NV_MPFS_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ib_routing_mode_supported		sriv_ib_routing_mode_supported	unknown_uplink_mac_flood_supported																													00h

Table 561 - NV_MPFS_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	ib_routing_mode_supported	BOOL	When set, IB_ROUTING_MODE is supported.
	29	sriv_ib_routing_mode_supported	BOOL	Set when SRIOV_IB_ROUTING_MODE is configurable
	28	unknown_uplink_mac_flood_supported	BOOL	When set, UNKNOWN_UPLINK_MAD_FLOOD is supported

11.7.94NV_POWER_CONF

Table 562 - NV_POWER_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
advanced_power_settings_en	slot_power_limiter_dis	reset_with_host_on_errors	sw_recovery_on_error	continuous_link_tuning	must not use	partial reset																										00h

Table 563 - NV_POWER_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	advanced_power_settings_en	BOOL	Show/hide additional power settings parameters. Warning: Wrong power settings may cause physical damage. mlxconfig_name: ADVANCED_POWER_SETTINGS
	30	slot_power_limiter_dis	BOOL	When cleared, the device is not allowed to consume more than 25W from the PCIe power rails, unless a PCI slot power limit message states that a new power limit is received. When set, the slot power limiter is disabled, and the device is allowed to consume more than 25W from the PCIe power rails. Note: if the power limiter is active and there is not enough power, the device will shut down the network modules. mlxconfig_name: DISABLE_SLOT_POWER_LIMITER
	29	reset_with_host_on_errors	BOOL	When set, host reset will trigger a device reset, if the device has previously detected fatal errors. [Internal] This parameter is not hidden by advanced_power_settings_en. mlxconfig_name: RESET_WITH_HOST_ON_ERRORS
	28	sw_recovery_on_error	BOOL	When set, SW will be instructed to perform a recovery flow when health buffer reports an error. [Internal] This parameter is hidden by advanced_power_settings_en. mlxconfig_name: SW_RECOVERY_ON_ERRORS
	27	continuous_link_tuning	BOOL	When set, the device will keep tuning the physical link after the link was raised
	26	must_not_use	BOOL	
	25	partial_reset	BOOL	When set, the partial reset flow is enabled. This reset skips on resetting GPIOs, loading PLL new parameters and performing self-tests. Partial reset will be triggered only if no exceptions requiring full reset were detected (for example, if any of the boot record's parameter had changed) mlxconfig_name: PARTIAL_RESET_EN

11.7.95 NV_POWER_CAP

Table 564 - NV_POWER_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
		slot power limiter supported	reset with host on errors supported	sw recovery on error supported	continuous link tuning supported	must not use	partial reset supported																									00h

Table 565 - NV_POWER_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	30	slot_power_limiter_supported	BOOL	When set, DISABLE_SLOT_POWER_LIMITER is configurable
	29	reset_with_host_on_errors_supported	BOOL	Indicate support for NV_POWER_CONF.reset_with_host_on_errors (RESET_WITH_HOST_ON_ERRORS)
	28	sw_recovery_on_error_supported	BOOL	Indicate support for NV_POWER_CONF.sw_recovery_on_error (SW_RECOVERY_ON_ERRORS)
	27	continuous_link_tuning_supported	BOOL	Indicates support for NV_POWER_CONF.continuous_link_tuning
	26	must_not_use	BOOL	
	25	partial_reset_supported	BOOL	When set, the partial reset flow is supported

11.7.96NV_HOST_SYNC_CONF

Table 566 - NV_HOST_SYNC_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
icmd_sem_lock_threshold																idle_icmd_sem_lock_threshold																00h
minimal_vsec_gw_expel_duration																																04h
																																08h-1Ch

Table 567 - NV_HOST_SYNC_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:16	icmd-sem-lock-threshold	UNSIGNED	Host will be expelled from VSEC Gateway usage if it holds the ICMD Global Semaphore for a period longer than this threshold. Value is given in units of 0.1sec Value 0xFFFF indicates this threshold is infinite (disabled) mlxconfig_name: ICMD_SEM_LOCK_THRESHOLD
	15:0	idle_icmd-sem-lock-threshold	UNSIGNED	Host will be expelled from VSEC Gateway usage if it holds the ICMD Global Semaphore, without issuing any ICMD, for a period longer than this threshold. Value is given in units of 0.1sec Value 0xFFFF indicates this threshold is infinite (disabled) mlxconfig_name: IDLE_ICMD_SEM_LOCK_THRESHOLD
04h	31:16	minimal_vsec_gw_expel_duration	UNSIGNED	Host VSEC Gateway expulsion will persist for this period. Value is given in units of 0.1sec mlxconfig_name: VSEC_EXPULSION_DURATION

11.7.97 NV_EMULATION_NVME_CONF

Table 568 - NV_EMULATION_NVME_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nvme_emu_enable																																00h
									nvme_emu_num_pf										nvme_emu_total_vf												04h	
nvme_emu_device_id															nvme_emu_vendor_id																08h	
~~nvme_emu_class_code~~ nvme_emu_revision_id									nvme_emu_class_code																					0Ch		
nvme_emu_subsystem_id															nvme_emu_subsystem_vendor_id																10h	
																					nvme_emu_num_msix										14h	
																																18h-1Ch

Table 569 - NV_EMULATION_NVME_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	nvme_emu_enable	BOOL	When set to TRUE, NVME device emulation is enabled. mlxconfig_name: NVME_EMULATION_ENABLE
04h	23:16	nvme_emu_num_pf	UNSIGNED	Total number of PCIe functions (PFs) exposed by the device for NVME emulation. mlxconfig_name: NVME_EMULATION_NUM_PF
	15:0	nvme_emu_total_vf	UNSIGNED	The total number of Virtual Functions (VFs) that can be supported for each PF. mlxconfig_name: NVME_EMULATION_NUM_VF
08h	31:16	nvme_emu_device_id	UNSIGNED	PCIe device_id config register for the NVME emulated device. mlxconfig_name: NVME_EMULATION_DEVICE_ID
	15:0	nvme_emu_vendor_id	UNSIGNED	PCIe vendor_id config register for the NVME emulated device. mlxconfig_name: NVME_EMULATION_VENDOR_ID

Table 569 - NV_EMULATION_NVME_CONF Field Descriptions

Offset	Bits	Name	Type	Description
0Ch	31:24	nvme_emu_revision_id	UNSIGNED	PCIe revision_id register for the NVME emulated device mlxconfig_name: NVME_EMULATION_REVISION_ID
	23:0	nvme_emu_class_code	UNSIGNED	PCIe class_code register for the NVME emulated device mlxconfig_name: NVME_EMULATION_CLASS_CODE
10h	31:16	nvme_emu_subsystem_id	UNSIGNED	PCIe subsystem_id register for the NVME emulated device mlxconfig_name: NVME_EMULATION_SUBSYSTEM_ID
	15:0	nvme_emu_subsystem_vendor_id	UNSIGNED	PCIe subsystem_vendor_id register for the NVME emulated device mlxconfig_name: NVME_EMULATION_SUBSYSTEM_VENDOR_ID
14h	15:0	nvme_emu_num_msix	UNSIGNED	Number of MSI-X Vectors assigned for the NVME emulation device. Value 0x0 will use device defaults mlxconfig_name: NVME_EMULATION_NUM_MSIX

11.7.98 NV_EMULATION_NVME_CAP

Table 570 - NV_EMULATION_NVME_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nvme_emu_supported																												00h				
									nvme_emu_max_num_pf								nvme_emu_max_total_vf													04h		
																																08h-1Ch

Table 571 - NV_EMULATION_NVME_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	nvme_emu_supported	BOOL	When set to TRUE, NVME device emulation is supported.
04h	23:16	nvme_emu_max_num_pf	UNSIGNED	Maximal number of PCIe physical functions (PFs) supported by the device for NVME emulation.
	15:0	nvme_emu_max_total_vf	UNSIGNED	Maximal number of PCIe virtual functions (VFs) supported by the device for each NVME emulation PF.

11.7.99 NV_EMULATION_PCI_SWITCH_CONF

Table 572 - NV_EMULATION_PCI_SWITCH_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
pci_switch																									num_switch_port								00h
																																04h	

Table 573 - NV_EMULATION_PCI_SWITCH_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	pci_switch	BOOL	When TRUE, the device will expose a PCI switch. All PF configurations are applied on the switch downstream ports. On such case, each PF will have a different PCI device on the emulated switch. This configuration allows to expose 0 network PFs toward host mlxconfig_name: PCI_SWITCH_EMULATION_ENABLE
	7:0	num_switch_port	UNSIGNED	Number of emulated switch ports. Each port can hold a single PCI device (emulated or not). mlxconfig_name: PCI_SWITCH_EMULATION_NUM_PORT

11.7.100NV_EMULATION_PCI_SWITCH_CAP

Table 574 - NV_EMULATION_PCI_SWITCH_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pci_switch_supported																																00h
																																04h

Table 575 - NV_EMULATION_PCI_SWITCH_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	pci_switch_supported	BOOL	When TRUE, PCI Switch Emulation is supported
	7:0	max_num_switch_port	UNSIGNED	Maximal number supported for emulated switch ports.

11.7.101NV_EMULATION_VIRTIO_NET_CONF

Table 576 - NV_EMULATION_VIRTIO_NET_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtio_net_emu_enable																																00h
																																04h
																																08h

Table 576 - NV_EMULATION_VIRTIO_NET_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtio_net_emu_revision_id								virtio_net_emu_class_code																				0Ch				
virtio_net_emu_subsystem_id															virtio_net_emu_subsystem_vendor_id												10h					
																virtio_net_emu_num_msix												14h				
																															18h-1Ch	

Table 577 - NV_EMULATION_VIRTIO_NET_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtio_net_emu_enable	BOOL	When set to TRUE, VIRTIO_NET device emulation is enabled. mlxconfig_name: VIRTIO_NET_EMULATION_ENABLE
04h	23:16	virtio_net_emu_num_pf	UNSIGNED	Total number of PCIe functions (PFs) exposed by the device for VIRTIO_NET emulation. mlxconfig_name: VIRTIO_NET_EMULATION_NUM_PF
	15:0	virtio_net_emu_total_vf	UNSIGNED	The total number of Virtual Functions (VFs) that can be supported for each PF. mlxconfig_name: VIRTIO_NET_EMULATION_NUM_VF
08h	31:16	virtio_net_emu_device_id	UNSIGNED	PCIe device_id config register for the VIRTIO_NET emulated device. mlxconfig_name: VIRTIO_NET_EMULATION_DEVICE_ID
	15:0	virtio_net_emu_vendor_id	UNSIGNED	PCIe vendor_id config register for the VIRTIO_NET emulated device. mlxconfig_name: VIRTIO_NET_EMULATION_VENDOR_ID
0Ch	31:24	virtio_net_emu_revision_id	UNSIGNED	PCIe revision_id register for the VIRTIO_NET emulated device mlxconfig_name: VIRTIO_NET_EMULATION_REVISION_ID
	23:0	virtio_net_emu_class_code	UNSIGNED	PCIe class_code register for the VIRTIO_NET emulated device mlxconfig_name: VIRTIO_NET_EMULATION_CLASS_CODE

Table 577 - NV_EMULATION_VIRTIO_NET_CONF Field Descriptions

Offset	Bits	Name	Type	Description
10h	31:16	virtio_net_emu_subsystem_id	UNSIGNED	PCIe subsystem_id register for the VIRTIO_NET emulated device mlxconfig_name: VIRTIO_NET_EMULATION_SUBSYSTEM_ID
	15:0	virtio_net_emu_subsystem_vendor_id	UNSIGNED	PCIe subsystem_vendor_id register for the VIRTIO_NET emulated device mlxconfig_name: VIRTIO_NET_EMU_SUBSYSTEM_VENDOR_ID
14h	15:0	virtio_net_emu_num_msix	UNSIGNED	Number of MSI-X Vectors assigned for the VIRTIO_NET emulation device. Value 0x0 will use device defaults mlxconfig_name: VIRTIO_NET_EMULATION_NUM_MSIX

11.7.102 NV_EMULATION_VIRTIO_NET_CAP

Table 578 - NV_EMULATION_VIRTIO_NET_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtio_net_emu_supported																														00h		
								virtio_net_emu_max_num_pf								virtio_net_emu_max_total_vf												04h				
																														08h-0Ch		

Table 579 - NV_EMULATION_VIRTIO_NET_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtio_net_emu_supported	BOOL	When set to TRUE, VIRTIO_NET device emulation is supported.
04h	23:16	virtio_net_emu_max_num_pf	UNSIGNED	Maximal number of PCIe physical functions (PFs) supported by the device for VIRTIO_NET emulation.
	15:0	virtio_net_emu_max_total_vf	UNSIGNED	Maximal number of PCIe virtual functions (VFs) supported by the device for each VIRTIO_NET emulation PF.

11.7.103 NV_EMULATION_VIRTIO_BLK_CONF

Table 580 - NV_EMULATION_VIRTIO_BLK_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtio_blk_emu_enable																														00h		
								virtio_blk_emu_num_pf								virtio_blk_emu_total_vf												04h				
virtio_blk_emu_device_id												virtio_blk_emu_vendor_id																08h				
virtio_blk_emu_revision_id								virtio_blk_emu_class_code																				0Ch				
virtio_blk_emu_subsystem_id												virtio_blk_emu_subsystem_vendor_id																10h				
																																14h
																												18h-1Ch				

Table 581 - NV_EMULATION_VIRTIO_BLK_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtio_blk_emu_enable	BOOL	When set to TRUE, VIRTIO_BLK device emulation is enabled. mlxconfig_name: VIRTIO_BLK_EMULATION_ENABLE
04h	23:16	virtio_blk_emu_num_pf	UNSIGNED	Total number of PCIe functions (PFs) exposed by the device for VIRTIO_BLK emulation. mlxconfig_name: VIRTIO_BLK_EMULATION_NUM_PF
	15:0	virtio_blk_emu_total_vf	UNSIGNED	The total number of Virtual Functions (VFs) that can be supported for each PF. mlxconfig_name: VIRTIO_BLK_EMULATION_NUM_VF
08h	31:16	virtio_blk_emu_device_id	UNSIGNED	PCIe device_id config register for the VIRTIO_BLK emulated device. mlxconfig_name: VIRTIO_BLK_EMULATION_DEVICE_ID
	15:0	virtio_blk_emu_vendor_id	UNSIGNED	PCIe vendor_id config register for the VIRTIO_BLK emulated device. mlxconfig_name: VIRTIO_BLK_EMULATION_VENDOR_ID
0Ch	31:24	virtio_blk_emu_revision_id	UNSIGNED	PCIe revision_id register for the VIRTIO_BLK emulated device mlxconfig_name: VIRTIO_BLK_EMULATION_REVISION_ID
	23:0	virtio_blk_emu_class_code	UNSIGNED	PCIe class_code register for the VIRTIO_BLK emulated device mlxconfig_name: VIRTIO_BLK_EMULATION_CLASS_CODE
10h	31:16	virtio_blk_emu_subsystem_id	UNSIGNED	PCIe subsystem_id register for the VIRTIO_BLK emulated device mlxconfig_name: VIRTIO_BLK_EMULATION_SUBSYSTEM_ID
	15:0	virtio_blk_emu_subsystem_vendor_id	UNSIGNED	PCIe subsystem_vendor_id register for the VIRTIO_BLK emulated device mlxconfig_name: VIRTIO_BLK_EMU_SUBSYSTEM_VENDOR_ID

Table 581 - NV_EMULATION_VIRTIO_BLK_CONF Field Descriptions

Offset	Bits	Name	Type	Description
14h	15:0	virtio_blk_emu_num_msix	UNSIGNED	Number of MSI-X Vectors assigned for the VIRTIO_BLK emulation device. Value 0x0 will use device defaults mlxconfig_name: VIRTIO_BLK_EMULATION_NUM_MSIX

11.7.104 NV_EMULATION_VIRTIO_BLK_CAP

Table 582 - NV_EMULATION_VIRTIO_BLK_CAP Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
virtio_blk_emu_supported																												00h					
								virtio_blk_emu_max_num_pf								virtio_blk_emu_max_total_vf																	04h
																												08h-0Ch					

Table 583 - NV_EMULATION_VIRTIO_BLK_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	virtio_blk_emu_supported	BOOL	When set to TRUE, VIRTIO_BLK device emulation is supported.
04h	23:16	virtio_blk_emu_max_num_pf	UNSIGNED	Maximal number of PCIe physical functions (PFs) supported by the device for VIRTIO_BLK emulation.
	15:0	virtio_blk_emu_max_total_vf	UNSIGNED	Maximal number of PCIe virtual functions (VFs) supported by the device for each VIRTIO_BLK emulation PF.

11.7.105 NV_VTS_BANK_A

Table 584 - NV_VTS_BANK_A Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fw_rev_major																																04h
fw_rev_minor																fw_rev_subminor																08h
ts_year																ts_month								ts_day								0Ch
ts_hour								ts_minute								ts_second																10h

Table 585 - NV_VTS_BANK_A Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:16	fw_rev_major	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A.
08h	31:16	fw_rev_minor	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A.
	15:0	fw_rev_subminor	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A
0Ch	31:16	ts_year	BINARY	Timestamp year - 4 packed BCD
	15:8	ts_month	BINARY	Timestamp month - 2 packed BCD
	7:0	ts_day	BINARY	Timestamp day - 2 packed BCD
10h	31:24	ts_hour	BINARY	Timestamp hour - 2 packed BCD
	23:16	ts_minute	BINARY	Timestamp minute - 2 packed BCD
	15:8	ts_second	BINARY	Timestamp second - 2 packed BCD

11.7.106 NV_VTS_BANK_B

Table 586 - NV_VTS_BANK_B Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fw_rev_major																																04h
fw_rev_minor																fw_rev_subminor																08h

Table 586 - NV_VTS_BANK_B Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ts_year																ts_month								ts_day								0Ch
ts_hour								ts_minute								ts_second																10h

Table 587 - NV_VTS_BANK_B Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:16	fw_rev_major	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A.
08h	31:16	fw_rev_minor	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A.
	15:0	fw_rev_sub-minor	UNSIGNED	The MVTS command use this parameter to keep the user defined version timestamp for FW bank A
0Ch	31:16	ts_year	BINARY	Timestamp year - 4 packed BCD
	15:8	ts_month	BINARY	Timestamp month - 2 packed BCD
	7:0	ts_day	BINARY	Timestamp day - 2 packed BCD
10h	31:24	ts_hour	BINARY	Timestamp hour - 2 packed BCD
	23:16	ts_minute	BINARY	Timestamp minute - 2 packed BCD
	15:8	ts_second	BINARY	Timestamp second - 2 packed BCD

Table 589 - NV_ROM_INI Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	undi_en	BOOL	Enable UNDI protocol
	30	hii_en	BOOL	Enable HII protocol
	29	plat-form_to_driver_en	BOOL	Enable Platform to Driver protocol
	28	bofm_en	BOOL	Enable BOFM protocol
	27	fmp_en	BOOL	Enable FW Management protocol
	26	flash_en	BOOL	Enable flash access
	25	port_enum	BOOL	Add port number to device name string
	24	mac_enum	BOOL	Add MAC address to device name string
	23	uefi_hii_over_ctrl	BOOL	When set to TRUE, UEFI driver should load UEFI HII protocols over the controller. Otherwise, UEFI driver should load UEFI HII protocols over the child. Valid if NV_ROM_CAP.uefi_proto_over_ctrl is TRUE
	22	uefi_fmp_over_ctrl	BOOL	When set to TRUE, UEFI driver should load UEFI FMP protocols over the controller. Otherwise, UEFI driver should load UEFI FMP protocols over the child. Valid if NV_ROM_CAP.uefi_proto_over_ctrl is TRUE
	21	uefi_aip_over_ctrl	BOOL	When set to TRUE, UEFI driver should load UEFI AIP protocols over the controller. Otherwise, UEFI driver should load UEFI AIP protocols over the child. Valid if NV_ROM_CAP.uefi_proto_over_ctrl is TRUE
	13	hii_read_only	BOOL	HII read only mode
	12	hii_bdf_decimal	BOOL	HII read only mode
	11	hii_platform_setup	BOOL	Use decimal values for PCIe address
	10	hii_ibm_aim	BOOL	Enable AIM HII
	9	hii_iscsi_configuration	BOOL	Enable ISCSI HII config
	8	static_component_name_string	BOOL	Use static device name
	3	tech_enum	BOOL	Add technology enumeration to device name string
	2	hii_vpi_en	BOOL	Enable HII VPI config
	1	shared_memory_en	BOOL	Enable HP shared memory functions

Table 589 - NV_ROM_INI Field Descriptions

Offset	Bits	Name	Type	Description
04h-08h	31:24	dhcp_user_class[8]	UNSIGNED	DHCP user class option (RFC3004) for Flexboot. Octet #1 must be equal for 77d. octet #2 - length
0Ch	31	promiscuous_vlan	BOOL	Set Flexboot to ignore incoming VLAN tags.
	30	option_rom_debug	BOOL	Enable/disable logs per the configured level for each module
	29:26	uri_boot_retry	UNSIGNED	Number of uri boot retries
	25:22	uri_boot_retry_delay	UNSIGNED	Delay in seconds between uri boot retries
	21	hii_aim_ucm_ver2	BOOL	Expose AIM ver, and UCM ver HII fields with value equal to 2. For Lenovo Agentless.
	20	hii_mriname2	BOOL	Expose MRI_NAME2 field in HII with hard coded value of "Mellanox Network Adapter, physical ports \$p, Logical ports 1" per Lenovo's HII spec
	19:16	hii_type	ENUM	Type of vfr type with HII configuration selector: 0x0: MLNX_DEFAULT 0x1: HPE 0x2: LENOVO 0x3: DELL 0x4: FUJITSU
	15	guid_ops	BOOL	guided_ops in CX3 (show specific fields in HII for Lenovo)
	14	hii_flexaddr_setting	BOOL	Flexaddress setting
	13	hii_flexaddr_override	BOOL	Override flexaddress displaying logic
	12	dhcp_pxe_discovery_control_dis	BOOL	Disable PXE DISCOVERY_CONTROL DHCP option support (option 43 sub option 6).
	11	tivoli_wa_en	BOOL	Enable (Tivoli) Rembo WA
	10	ucm_single_port	BOOL	When set, the mriname string will not include the port number. (relevant to Lenovo Purely cards)

11.8.2 NV_ROM_CAP

Table 590 - NV_ROM_CAP Layout

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
																						uefi proto over ctrl	boot legacy interrupt disable supported	legacy boot wo failover supported	uefi undi vlan config en	uefi undi config en	hii config en	lasp config en	uefi logs en	flexboot debug en	boot debug log en	boot ip_ver

Table 591 - NV_ROM_CAP Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	boot_ip_ver	BOOL	This bit is set when flexboot supports configuration of PXE boot IP version defined in boot settings ext parameter
	30	boot_de- bug_log_en	BOOL	
	29	flexboot_de- bug_en	BOOL	When set, Flexboot debug is supported (NV_ROM_FLEX-BOOT_DEBUG and NV_ROM_DEBUG_LEVEL are applicable)
	28	uefi_logs_en	BOOL	When set, UEFI logging is supported (NV_ROM_UEFI_DEBUG_LEVEL is applicable)
	27	lACP_con- fig_en	BOOL	When set, Flexboot supports configuration of LACP (NV_ROM_BOOT_CONF1.lACP_dis)
	26	hii_config_en	BOOL	When set, NV_ROM_UEFI_CONF.hii_en controls if HII is reported (overriding NV_ROM_INI.hii_en)
	25	uefi_undi_ - config_en	BOOL	When set, NV_ROM_UEFI_CONF.uefi_undi_en controls if UEFI UNDI is reported
	24	uefi_undi_vla n_config_en	BOOL	When set, mlxconfig and UEFI-HII menus shows the UEFI_UNDI_VLAN_EN field.
	23	legacy_ - boot_wo_ - failover_supp orted	BOOL	When set to TRUE, FlexBoot supports configuration of boot protocol, without failover to another protocol upon failure (NV_ROM_BOOT_CONF1.legacy_boot_protocol).
	22	boot_lega- cy_interrupt_ - disable_supp orted	BOOL	When set to TRUE, legacy interrupt usage can be disabled by boot_legacy_interrupt_disable
	21	uefi_pro- to_ - over_ctrl	BOOL	

11.8.3 NV_ROM_BOOT_CONF2

Table 592 - NV_ROM_BOOT_CONF2 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
									undi_network_wait														boot_ip_ver									00h

Table 593 - NV_ROM_BOOT_CONF2 Field Descriptions

Offset	Bits	Name	Type	Description
00h	23:16	undi_network_wait	UNSIGNED	The number of seconds Flexboot waits after issuing a UNDI open commands until it starts using the interface. mlxconfig_name: BOOT_UNDI_NETWORK_WAIT
	9:8	boot_ip_ver	ENUM	Select which IP protocol version will be used by Flexboot. If both are configured, Flexboot will try to boot with the 2nd protocol only if DHCP parameters for the first protocol are not available, or if booting with the first protocol has failed. [Internal] This field is supported if the 'nv_config - boot_ip_ver' field in 'OPTION ROM capabilities' is set to 1. 0x0: IPv4 0x1: IPv6 0x2: IPv4_IPv6 0x3: IPv6_IPv4 mlxconfig_name: IP_VER

11.8.4 NV_ROM_UEFI_CONF

Table 594 - NV_ROM_UEFI_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hii_en	uefi_undi_dis	uefi_undi_vlan_en														uefi_undi_vlan_id																00h

Table 595 - NV_ROM_UEFI_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	hii_en	BOOL	Enable UEFI/BIOS HII menus for this PCI function. Warning: HII menus might be required for fixing and debugging of system boot issues. [Internal] This bit is applicable only if NV_ROM_CAP.hii_config_en is set. mlxconfig_name: UEFI_HII_EN
	30	uefi_undi_dis	BOOL	Disable the UEFI expansion ROM support for the UNDI driver, which prevents the UEFI BIOS from using this PCI function for networking. This parameter is supported only if NV_ROM_CAP.uefi_undi_config_en is set.
	29	uefi_undi_vlan_en	BOOL	This parameter is only shown when NV_ROM_CAP.uefi_undi_vlan_config_en bit is set. mlxconfig_name: UEFI_UNDI_VLAN_EN mlxconfig_desc: When set, the UEFI UNDI driver will add VLAN tagging, for outgoing traffic, and will only accept incoming traffic on the configured VLAN ID. VLAN ID is configured by the UEFI_UNDI_VLAN_ID parameter. This feature should not be used if VLAN is configured in the BIOS.
	11:0	uefi_undi_vlan_id	BINARY	This parameter is only shown when uefi_undi_vlan_en is set. mlxconfig_name: UEFI_UNDI_VLAN_ID mlxconfig_desc: This field configure the VLAN ID used by the UEFI UNDI driver (per UEFI_UNDI_VLAN_EN).

11.8.5 NV_ROM_BOOT_CONF1

Table 596 - NV_ROM_BOOT_CONF1 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
boot_option_rom_en				lACP_dis		boot_vlan_en		boot_legacy_interrupt_disable		boot_retry_count		legacy_boot_protocol								boot_vlan												00h

Table 597 - NV_ROM_BOOT_CONF1 Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	boot_option_rom_en	BOOL	When TRUE, the device is recognized as boot device at POST stage
	30	boot_vlan_en	BOOL	Enable VLAN mode for network boot. mlxconfig_name: BOOT_VLAN_EN
	29	lacp_dis	BOOL	Disable IEEE 802.3ad Link Aggregation Control Protocol (LACP) for legacy BIOS expansion ROM (Flexboot). [Internal] This setting is supported when NV_ROM_CAP.lacp_config_en bit is set. mlxconfig_name: BOOT_LACP_DIS
	27	boot_legacy_interrupt_disable	BOOL	When set to TRUE, legacy interrupts should not be used for receive/transmit indication. Polling should be used instead. Supported only if boot_legacy_interrupt_disable_supported=TRUE. mlxconfig_name: BOOT_INTERRUPT_DIS
	26:24	boot_retry_count	ENUM	Number of retries to attempt in case of boot failure (Flex-Boot) 0x0: NONE 0x1: _1 0x2: _2 0x3: _3 0x4: _4 0x5: _5 0x6: _6 0x7: UNLIMITED mlxconfig_name: BOOT_RETRY_CNT supported_from_version: 1
	23:16	legacy_boot_protocol	ENUM	Select boot protocol for legacy BIOS expansion ROM (Flexboot) 0x0: NONE - The expansion ROM will not add this PCI PF as a boot target during the POST stage. 0x1: PXE - The expansion ROM will use PXE boot. 0x2: ISCSI - The expansion ROM will use iSCSI boot. 0x3: BOTH - [not supported] The expansion ROM will use PXE boot and iSCSI boot 0x4: PXE_WO_FAIL_ISCSI - PXE without fail-over to ISCSI. Supported when legacy_boot_wo_failover_supported==1. 0x5: ISCSI_WO_FAIL_PXE - ISCSI without fail-over to PXE. Supported when legacy_boot_wo_failover_supported==1. mlxconfig_name: LEGACY_BOOT_PROTOCOL
	11:0	boot_vlan	UNSIG	VLAN ID for network boot. mlxconfig_name: BOOT_VLAN

11.8.6 NV_ROM_IB_BOOT_CONF

Table 598 - NV_ROM_IB_BOOT_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																boot_pkey												00h				

Table 599 - NV_ROM_IB_BOOT_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	15:0	boot_pkey	UNSIGNED	Infiniband P_Key to be used by PXE boot mlxconfig_name: BOOT_PKEY

11.8.7 NV_ROM_ISCSI_GENERAL_CONF

Table 600 - NV_ROM_ISCSI_GENERAL_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
iscsi_keep_san_dev_en		chap_auth_en		chap_mutual_auth_en		tcp_timestamps_en		vlan_en				boot_to_target																00h				
																				vlan										04h		
									iscsi_drive_num						link_up_delay_time						lun_busy_retry_count						08h					

Table 601 - NV_ROM_ISCSI_GENERAL_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	30	iscsi_keep-san_dev_en	BOOL	Keep the iSCSI device attached for the OS supported_from_version: 2
	29	chap_auth_en	BOOL	Authenticate the initiator to the target using CHAP. xuefi: chapAuthEnable supported_from_version: 1
	28	chap_mutual_auth_en	BOOL	Authenticate the target to the initiator using CHAP. xuefi: chapMutualAuth supported_from_version: 1
	27	tcp_time-stamps_en	BOOL	Enable the TCP timestamps option for iSCSI TCP connections. xuefi: tcpTimestamp
	26	vlan_en	BOOL	Enable VLANs for iSCSI xuefi: iscsiVlanMode supported_from_version: 1
	23:22	boot_to_target	ENUM	Boot to iSCSI target after connection 0x0: ENABLED - Boot from target 0x1: DISABLED - Don't boot from target (target will be hooked if configured and iscsi_keep_san_dev_en is set) 0x2: ONE_TIME_DISABLED - Used for installing a new OS on the iSCSI target. Flexboot will not boot from the target once (e.g. DISABLED), and then will automatically change the value to ENABLED. xuefi: iscsiTgtBoot
04h	11:0	vlan	UNSIGNED	VLAN number. Only valid when vlan_en is 1 xuefi: iscsiVlanId supported_from_version: 1
08h	23:16	iscsi_drive_num	ENUM	The HDD number for the hooked iSCSI target in legacy BIOS int13 drive table. 0x80: HDD1 0x81: HDD2 0x82: HDD3 0x83: HDD4 supported_from_version: 2
	15:8	link_up_delay_time	UNSIGNED	Number of seconds to wait after the link comes up before sending packets over the network. xuefi: lnkUpDelayTime supported_from_version: 1
	7:0	lun_busy_retry_count	UNSIGNED	Number of connection attempts to make if the iSCSI target is busy. Valid values are between 0 and 60. xuefi: lunBusyRetryCnt supported_from_version: 1

11.8.8 NV_ROM_DEBUG_LEVEL

Table 602 - NV_ROM_DEBUG_LEVEL Layout

Offset	00h	04h	08h
0	debug_log_en	stp	tcp
1			
2			tcpip
3			
4		rom-prefix	ipv4
5			
6			ipv6
7			
8		dhcp v6	driver_settings
9			
10		arp	status_update
11			
12		neighbor	pxe_undi
13			
14			
15			
16	uri		
17	driver		
18			
19	nod-nic		
20	nodnic_cmd		
21			
22	nodnic_device		
23			
24	nodnic_port		
25			
26	netdevice		
27			
28	tftp		
29			
30	udp		
31			

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
00h	0	debug_log_en	BOOL	When set, Flexboot generates debug logs at the levels specified by the following parameters: BOOT_DBG_LOG_STP, BOOT_DBG_LOG_ROMPREFIX, BOOT_DBG_LOG_DHCP, BOOT_DBG_LOG_DHCP6, BOOT_DBG_LOG_ARP, BOOT_DBG_LOG_NEIGHBOR, BOOT_DBG_LOG_NDP, BOOT_DBG_LOG_URI, BOOT_DBG_LOG_IB_DRV, BOOT_DBG_LOG_DRV_CMD, BOOT_DBG_LOG_DRV_DEV, BOOT_DBG_LOG_DRV_PORT, BOOT_DBG_LOG_TFTP, BOOT_DBG_LOG_UDP, BOOT_DBG_LOG_TCP, BOOT_DBG_LOG_TCPIP, BOOT_DBG_LOG_IP, and BOOT_DBG_LOG_IPV6. mlxconfig_name: BOOT_DBG_LOG

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
04h	31:30	udp	ENUM	Debug level for Flexboot's UDP stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_UDP
	29:28	tftp	ENUM	Debug level for Flexboot's TFTP stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_TFTP
	27:26	netdevice	ENUM	Network Device Link status and traffic 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NETDEVICE
	25:24	nodnic_port	ENUM	Debug level for Flexboot's NODNIC Driver port management. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NDRV_PORT
	23:22	nodnic_device	ENUM	Debug level for Flexboot's NODNIC Driver tear-down and bring ups operations. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NDRV_DEV

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
04h	21:20	nodnic_cmd	ENUM	Debug level for Flexboot's NODNIC Driver interface 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NDRV_CMD
	19:18	nodnic	ENUM	Debug level for Flexboot's NODNIC driver (init, teardown, poll CQ etc) 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NDRV
	17:16	driver	ENUM	Debug level for Flexboot's driver (init, teardown, poll CQ etc) 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_DRV
	15:14	uri	ENUM	Debug level for Flexboot's URI parsing code 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_URI
	13:12	ndp	ENUM	Debug level for Flexboot's IPv6 Neighbor Discovery Protocol 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NDP

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
04h	11:10	neighbor	ENUM	Debug level for Flexboot's Neighbor table (which is filled by ARP and NDP) 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_NEIGHBOR
	9:8	arp	ENUM	Debug level for Flexboot's Address resolution Protocol (ARP) 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_ARP
	7:6	dhcpv6	ENUM	Debug level for Flexboot's DHCPv6 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_DHCP6
	5:4	dhcp	ENUM	Debug level for Flexboot's DHCP 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_DHCP
	3:2	romprefix	ENUM	Debug level for Flexboot's driver bootstrap code 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_ROMPREFIX
	1:0	stp	ENUM	Debug level for Flexboot's Spanning Tree Protocol 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_STP

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
08h	13:12	pxe_undi	ENUM	Debug level for Flexboot's UNDI interface 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_PXE_UNDI
	11:10	status_update	ENUM	Debug level for status update interface. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_STATUS
	9:8	driver_settings	ENUM	Debug level for non volatile configuration management. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_DRIVER_SETTINGS
	7:6	ipv6	ENUM	Debug level for Flexboot's IPv6 stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_IPV6
	5:4	ipv4	ENUM	Debug level for Flexboot's IPv4 stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_IP
	3:2	tcpip	ENUM	Debug level for Flexboot's TCP/IP stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_TCPIP

Table 603 - NV_ROM_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
08h	1:0	tcp	ENUM	Debug level for Flexboot's TCP stack. 0x0: DISABLE 0x1: BASIC 0x2: ADVANCED 0x3: ALL mlxconfig_name: BOOT_DBG_LOG_TCP

11.8.9 NV_ROM_UEFI_DEBUG_LEVEL

Table 604 - NV_ROM_UEFI_DEBUG_LEVEL Layout

Offset	00h																														
0	uefi_debug_log_init																														
1	uefi_debug_log_load																														
2	uefi_debug_log_fs																														
3	uefi_debug_log_pool																														
4	uefi_debug_log_page																														
5	uefi_debug_log_dispatch																														
6	uefi_debug_log_bm																														
7	uefi_debug_log_blkio																														
8	uefi_debug_log_net																														
9	uefi_debug_log_undi																														
10	uefi_debug_log_event																														
11	uefi_debug_log_gcd																														
12	uefi_debug_log_cache																														
13																															
14																															
15																															
16																															
17																															
18																															
19																															
20																															
21																															
22																															
23	uefi_log_severity																														
24																															
25																															
26																															
27	uefi_logs_verbose																														
28	uefi_logs																														
29																															
30																															
31																															

Table 605 - NV_ROM_UEFI_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
00h	31:28	uefi_logs	ENUM	Select the output device logs generated by the NIC/HBA UEFI expansion ROM. 0x0: DISABLED 0x1: STDOUT 0x2: STDERR mlxconfig_name: UEFI_LOGS
	27	uefi_logs_verbose	BOOL	Detailed debug messages that may significantly impact boot performance debug messages. mlxconfig_name: UEFI_LOG_VERBOSE
	25:24	uefi_log_severity	ENUM	NIC/HBA UEFI expansion ROM 0x0: ERROR 0x1: WARNING 0x2: INFO 0x3: DEBUG Note - Enabling of debug logs for specific modules is possible after setting the severity to DEBUG. mlxconfig_name: UEFI_LOG_MIN_SEVERITY
	12	uefi_debug_log_cache	BOOL	Enable UEFI debug level logs for Memory range cacheability changes. mlxconfig_name: UEFI_DEBUG_LOG_CACHE
	11	uefi_debug_log_gcd	BOOL	Enable UEFI debug level logs for Global Coherency Database changes. mlxconfig_name: UEFI_DEBUG_LOG_GCD
	10	uefi_debug_log_event	BOOL	Enable UEFI debug level logs of Event. mlxconfig_name: UEFI_DEBUG_LOG_EVENTS

Table 605 - NV_ROM_UEFI_DEBUG_LEVEL Field Descriptions

Offset	Bits	Name	Type	Description
00h	9	uefi_debug_log_undi	BOOL	Enable UEFI debug level logs UNDI Driver. mlxconfig_name: UEFI_DEBUG_LOG_UNDI
	8	uefi_debug_log_net	BOOL	Enable UEFI debug level logs SNI Driver. mlxconfig_name: UEFI_DEBUG_LOG_SNI
	7	uefi_debug_log_blkio	BOOL	Enable UEFI debug level logs for BlkIo Driver. mlxconfig_name: UEFI_DEBUG_LOG_BLKIO
	6	uefi_debug_log_bm	BOOL	Enable UEFI debug level logs for Boot Manager. mlxconfig_name: UEFI_DEBUG_LOG_BM
	5	uefi_de- bug_log_dis- patch	BOOL	Enable UEFI debug level logs for PEI/DXE/SMM Dis- patchers. mlxconfig_name: UEFI_DEBUG_LOG_DISPATCH
	4	uefi_debug_log_page	BOOL	Enable UEFI debug level logs for page Alloc & Free. mlxconfig_name: UEFI_DEBUG_LOG_PAGE
	3	uefi_de- bug_log_pool	BOOL	Enable UEFI debug level logs for pool Alloc & Free. mlxconfig_name: UEFI_DEBUG_LOG_POOL
	2	uefi_de- bug_log_fs	BOOL	Enable UEFI debug level logs for EFI file system accesses. mlxconfig_name: UEFI_DEBUG_LOG_FS
	1	uefi_de- bug_log_load	BOOL	Enable UEFI debug level logs for load events mlxconfig_name: UEFI_DEBUG_LOG_LOAD
	0	uefi_de- bug_log_init	BOOL	Enable UEFI debug level initialization log. mlxconfig_name: UEFI_DEBUG_LOG_INIT

11.8.10 NV_ROM_FLEXBOOT_DEBUG

Table 606 - NV_ROM_FLEXBOOT_DEBUG Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
boot_to_shell		panic_banner																														00h

Table 607 - NV_ROM_FLEXBOOT_DEBUG Field Description

Offset	Bits	Name	Type	Description
00h	31	boot_to_shell	BOOL	When this bit is set, Flexboot will boot to ipxe shell instead of the normal boot.
	30:29	panic_banner	ENUM	Flexboot behavior on critical failure 0x0: PRINT - print error to screen 0x1: HALT - freeze system 0x2: OPEN_SHELL - open ipxe shell

11.8.11 NV_ROM_BANNER_TIMEOUT

Table 608 - NV_ROM_BANNER_TIMEOUT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												flexboot_banner_timeout	00h			

Table 609 - NV_ROM_BANNER_TIMEOUT Field Description

Offset	Bits	Name	Type	Description
00h	3:0	flexboot_banner_timeout	UNSIGNED	Number of seconds to wait when displaying the timer for the boot menu key-press. If set to 0 the boot menu is disabled. The boot menu is only allowed to write values between 1 and 15. UEFI and mlxconfig may write 0.

11.8.12 NV_ROM_OCDS_OCBB

Table 610 - NV_ROM_OCSD_OCBB Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ocsd_en																															00h	

Table 611 - NV_ROM_OCSD_OCBB Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	ocsd_en	BOOL	Enable HP on-chip temperature monitoring (OCSD) and Active Health Systems support via FW (OCBB) reporting.

11.8.13 NV_ROM_ISCSI_DHCP_CONF

Table 612 - NV_ROM_ISCSI_DHCP_CONF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ipv4_dhcp_en	dhcp_iscsi_en																															00h

Table 613 - NV_ROM_ISCSI_DHCP_CONF Field Descriptions

Offset	Bits	Name	Type	Description
00h	31	ipv4_dhcp_en	BOOL	When enabled, Flexboot attempts to discover IPv4 configuration for iSCSI through DHCP. xuefi: TcpIpViaDhcp
	30	dhcp_iscsi_en	BOOL	When enabled, Flexboot attempts to discover iSCSI parameters through DHCP. xuefi: IscsiViaDhcp

11.8.14 NV_ROM_ISCSI_DHCP_VENDOR_ID

Table 614 - NV_ROM_ISCSI_DHCP_VENDOR_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																::
value[252]								value[253]								value[254]								value[255]								FCh

Table 615 - NV_ROM_ISCSI_DHCP_VENDOR_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h-FCh	31:24	value[256]	STRING	Flexboot DHCP Vendor ID xuefi: DhcpVendId

11.8.15 NV_ROM_ISCSI_INITIATOR_IP_ADDR

Table 616 - NV_ROM_ISCSI_INITIATOR_IP_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[12]								value[13]								value[14]								value[15]								0Ch

Table 617 - NV_ROM_ISCSI_INITIATOR_IP_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	iSCSI initiator IPv4 address in dotted decimal notation (Flexboot) xuefi: IscsiInitiatorIpv4Addr

11.8.16 NV_ROM_ISCSI_INITIATOR_SUBNET_MASK

Table 618 - NV_ROM_ISCSI_INITIATOR_SUBNET_MASK Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[12]								value[13]								value[14]								value[15]								0Ch

Table 619 - NV_ROM_ISCSI_INITIATOR_SUBNET_MASK Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	iSCSI initiator IPv4 subnet mask in dotted decimal notation (Flexboot) xuefi: IscsiInitiatorSubnet

11.8.17 NV_ROM_ISCSI_INITIATOR_GATEWAY

Table 620 - NV_ROM_ISCSI_INITIATOR_GATEWAY Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																::
value[12]								value[13]								value[14]								value[15]								0Ch

Table 621 - NV_ROM_ISCSI_INITIATOR_GATEWAY Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	IPv4 gateway for the iSCSI initiator in dotted decimal notation (Flexboot) xuefi: IscsiInitiatorIpv4Gateway

11.8.18 NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR

Table 622 - NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																::
value[12]								value[13]								value[14]								value[15]								0Ch

Table 623 - NV_ROM_ISCSI_INITIATOR_PRIMARY_DNS_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	Primary IPv4 DNS for iSCSI initiator in dotted decimal notation (Flexboot) xuefi: IscsiInitiatorIpv4PrimDns

11.8.19 NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR

Table 624 - NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[12]								value[13]								value[14]								value[15]								0Ch

Table 625 - NV_ROM_ISCSI_INITIATOR_SECONDARY_DNS_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	Secondary IPv4 DNS for iSCSI initiator in dotted decimal notation. (Flexboot) xuefi: IscsiInitiatorIpv4SecDns

11.8.20 NV_ROM_ISCSI_INITIATOR_NAME

Table 626 - NV_ROM_ISCSI_INITIATOR_NAME Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[220]								value[221]								value[222]								value[223]								DCh

Table 627 - NV_ROM_ISCSI_INITIATOR_NAME Field Descriptions

Offset	Bits	Name	Type	Description
00h-DCh	31:24	value[224]	STRING	This configuration sets the initiator name for iSCSI boot. According to the iSCSI RFC (3270) the name has to be UTF-8 encoded, with total length that does not exceed 223 bytes. It allows international characters, but prohibits the use of invisible characters, such as whitespace characters. (Flexboot) xuefi: IscsiInitiatorName

11.8.21 NV_ROM_ISCSI_INITIATOR_CHAP_ID

Table 628 - NV_ROM_ISCSI_INITIATOR_CHAP_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[124]								value[125]								value[126]								value[127]								7Ch

Table 629 - NV_ROM_ISCSI_INITIATOR_CHAP_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h-7Ch	31:24	value[128]	STRING	CHAP identifier (user name) for iSCSI initiator (Flexboot) xuefi: IscsiInitiatorChapId

11.8.22 NV_ROM_ISCSI_INITIATOR_CHAP_PWD

Table 630 - NV_ROM_ISCSI_INITIATOR_CHAP_PWD Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[12]								value[13]								value[14]								value[15]								0Ch

Table 631 - NV_ROM_ISCSI_INITIATOR_CHAP_PWD Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	CHAP secret (password) for the initiator (Flexboot) xuefi: IscsiInitiatorChapPwd

11.8.23 NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET

Table 632 - NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h

Table 633 - NV_ROM_ISCSI_CONNECT_TO_FIRST_TARGET Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	value[8]	STRING	When the string is "Enabled", Flexboot attempts to connect to the first iSCSI target. Other legal value is "Disabled"

11.8.24 NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET

Table 634 - NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h

Table 635 - NV_ROM_ISCSI_CONNECT_TO_SECOND_TARGET Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	value[8]	STRING	When the string is "Enabled", Flexboot attempts to connect to the first iSCSI target. Other legal value is "Disabled"

11.8.25 NV_ROM_ISCSI_FIRST_TARGET_ADDR

Table 636 - NV_ROM_ISCSI_FIRST_TARGET_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[12]								value[13]								value[14]								value[15]								0Ch

Table 637 - NV_ROM_ISCSI_FIRST_TARGET_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	IPv4 address of the first iSCSI target in dotted decimal notation (Flexboot)

11.8.26 NV_ROM_ISCSI_SECOND_TARGET_ADDR

Table 638 - NV_ROM_ISCSI_SECOND_TARGET_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h

Table 638 - NV_ROM_ISCSI_SECOND_TARGET_ADDR Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[12]								value[13]								value[14]								value[15]								0Ch

Table 639 - NV_ROM_ISCSI_SECOND_TARGET_ADDR Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	IPv4 address of the second iSCSI target in dotted decimal notation (Flexboot)

11.8.27 NV_ROM_ISCSI_FIRST_TARGET_PORT

Table 640 - NV_ROM_ISCSI_FIRST_TARGET_PORT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h

Table 641 - NV_ROM_ISCSI_FIRST_TARGET_PORT Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	value[8]	STRING	TCP port for the first iSCSI target (Flexboot) Decimal integer in ASCII format up to 5 digits (1-65536).

11.8.28 NV_ROM_ISCSI_SECOND_TARGET_PORT

Table 642 - NV_ROM_ISCSI_SECOND_TARGET_PORT Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h

Table 643 - NV_ROM_ISCSI_SECOND_TARGET_PORT Field Descriptions

Offset	Bits	Name	Type	Description
00h-04h	31:24	value[8]	STRING	TCP port for the second iSCSI target (Flexboot) Decimal integer in ASCII format up to 5 digits (1-65536).

11.8.29 NV_ROM_ISCSI_FIRST_TARGET_LUN

Table 644 - NV_ROM_ISCSI_FIRST_TARGET_LUN Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[16]								value[17]								value[18]								value[19]								10h

Table 645 - NV_ROM_ISCSI_FIRST_TARGET_LUN Field Descriptions

Offset	Bits	Name	Type	Description
00h-10h	31:24	value[20]	STRING	ISCSI first target boot LUN (Flexboot) Decimal integer 1- 18,446,744,073,709,551,615

11.8.30 NV_ROM_ISCSI_FIRST_TARGET_LUN

Table 646 - NV_ROM_ISCSI_SECOND_TARGET_LUN Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																..
value[16]								value[17]								value[18]								value[19]								10h

Table 647 - NV_ROM_ISCSI_SECOND_TARGET_LUN Field Descriptions

Offset	Bits	Name	Type	Description
00h-10h	31:24	value[20]	STRING	ISCSI second target boot LUN (Flexboot) Decimal integer 1- 18,446,744,073,709,551,615

11.8.31 NV_ROM_ISCSI_FIRST_TARGET_NAME

Table 648 - NV_ROM_ISCSI_FIRST_TARGET_NAME Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[220]								value[221]								value[222]								value[223]								DCh

Table 649 - NV_ROM_ISCSI_FIRST_TARGET_NAME Field Descriptions

Offset	Bits	Name	Type	Description
00h-DCh	31:24	value[224]	STRING	First iSCSI target name (Flexboot) up to 223 characters.

11.8.32 NV_ROM_ISCSI_SECOND_TARGET_NAME

Table 650 - NV_ROM_ISCSI_SECOND_TARGET_NAME Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[220]								value[221]								value[222]								value[223]								DCh

Table 651 - NV_ROM_ISCSI_SECOND_TARGET_NAME Field Descriptions

Offset	Bits	Name	Type	Description
00h-DCh	31:24	value[224]	STRING	Second iSCSI target name (Flexboot) up to 223 characters.

11.8.33 NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID

Table 652 - NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[124]								value[125]								value[126]								value[127]								7Ch

Table 653 - NV_ROM_ISCSI_FIRST_TARGET_CHAP_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h-7Ch	31:24	value[128]	STRING	iSCSI first target username/id (Flexboot)

11.8.34 NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID

Table 654 - NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[124]								value[125]								value[126]								value[127]								7Ch

Table 655 - NV_ROM_ISCSI_SECOND_TARGET_CHAP_ID Field Descriptions

Offset	Bits	Name	Type	Description
00h-7Ch	31:24	value[128]	STRING	iSCSI second target username/id (Flexboot)

11.8.35 NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD

Table 656 - NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[12]								value[13]								value[14]								value[15]								0Ch

Table 657 - NV_ROM_ISCSI_FIRST_TARGET_CHAP_PWD Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	iSCSI first target password (Flexboot)

11.8.36 NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD

Table 658 - NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
value[0]								value[1]								value[2]								value[3]								00h
value[4]								value[5]								value[6]								value[7]								04h
...																																...
value[12]								value[13]								value[14]								value[15]								0Ch

Table 659 - NV_ROM_ISCSI_SECOND_TARGET_CHAP_PWD Field Descriptions

Offset	Bits	Name	Type	Description
00h-0Ch	31:24	value[16]	STRING	iSCSI second target password (Flexboot)

11.9 Internal NIC Parameters

11.9.1 NV_IDX_HM_RESTORE_PARAMS

This internal TLV is used by the NIC to preserve host management parameters across reboots in specific environments.

The internal structure of this TLV is undefined.

12 No-DRAM NIC

12.1 Introduction

No DRAM NIC (NODNIC) is a lightweight interface intended to allow simple packet transmission for minimalistic drivers with a low memory footprint. It supports Ethernet and InfiniBand UD packet transmission and reception. The interface is handled through a copy of the “Initialization Segment” that is accessible using configuration cycles. The Mellanox “VSC - Vendor Specific Capability” in the PCI configuration space allows accessing the initialization segment. Inside this segment, a special section holds NODNIC configuration registers, as defined in “No DRAM NIC Configuration Registers”. The VSC is the only method for accessing the NODNIC registers in the initialization segment, unless specified otherwise.

Although the control of the NODNIC interface can be done using PCI configuration cycles, packet transmission and reception is done using DMA - in a similar way to the regular driver.

Accessing the initialization segment over VSC and PCI configuration cycles may be relatively slow. For improved performance, the NODNIC has an option to perform all packet transmission and reception control operations directly over the BAR (while keeping the control interface over VSC).

12.2 Limitations

The No DRAM interface does not support virtualization. A physical function working in this mode will not support virtual functions on top of it, and virtual functions will not operate in NODNIC mode unless `NV_GLOBAL_PCI_CONF.vf_nodnic_enable` is set to TRUE.

NODNIC over an Ethernet port has a single transmit ring and a single receive ring (ring0). NodNIC over an IB port has 2 QPs (each built by a receive ring and a transmit ring). Ring0 is used for IB QP1, and ring1 is a general purpose UD QP.

12.3 Initialization and Teardown

A driver initializing the no-DRAM NIC interface initializes the device and then initializes the port according to the sections below. Unless specified otherwise, all configuration registers refer to the registers in the “Initialization Segment”.

12.3.1 Device Initialization

1. Wait for *initializing* to be cleared
2. Initiate Function Level Reset (FLR)
3. Wait for *initializing* to be cleared
4. Verify *nic_interface_supported[2]* (NODNIC) is set.
5. Write *nic_interface* = 2 (no-DRAM NIC).
6. Wait for *initializing* to be cleared.
7. In PCI command register, set the bus master bit and clear the interrupt disable bit.
8. Read *no_dram_nic_offset* field to find where the NODNIC registers are located.

9. Verify `no_dram_nic_revision = 1`.
10. Read NODNIC parameters: device capabilities, hardware format, CQE format, etc.

12.3.2 Port Initialization

1. Read the `event` word in order to clear it.
2. Read the `link_type` to determine whether the port is IB or Ethernet. For IB, read `P_Key` table and the port parameters (GID, LID, SM SL, and SM LID). For Ethernet, read the hardware MAC address.
3. For each applicable ring, fill out the parameters: `queue_address`, `log_size`, IB parameters if needed (`qkey`, `pkey_index`).
 - a. The ring size must not exceed `log_max_ring_size`.
4. Fill out port registers:
 - a. Read completion queue size in `log_cq_size`, and allocate memory for it.
 - b. Fill the completion queue address in `completion_address`.
 - c. Read the working buffer size from `log_working_buffer_size` and allocate memory for it.
 - d. Fill the working buffer address in `working_buffer_address`.
5. If the driver wishes to use Rx doorbell record (see “Doorbell Record” on page 308), set the `db_record_addr_h/l` in the Rx ring parameters. The support for Rx doorbell record for NODNIC driver is indicated by `support_db_record_rx` in “General Configurations” on page 815
6. If the driver wishes to use Tx doorbells through UAR (see “User Access Region (UAR)” on page 249), set the `db_record_addr_h/l` in the Tx ring parameters, and zero the doorbell record. The support for Tx doorbell through UAR for NODNIC driver is indicated by `support_uar_tx_doorbell` in “General Configurations” on page 815
7. Activate networking by writing `network_en = 1`.
 - a. Optionally activate DMA, see “Enabling/Disabling DMA” on page 827.
8. Set receive filters as described in “Setting Receive Filters”.
9. Check `port_state` to determine the logical link state.

12.3.3 Teardown

The teardown process of the no-DRAM NIC interface is comprised of the initialization process in reverse order.

1. Disablenetworking of the port:
 - a. Disable DMA, see “Enabling/Disabling DMA” on page 827.
 - b. Write `network_en = 0`.
2. Once `network_en` is cleared, allocated resources for the port can be released.
3. Write `nic_interface = 1` (disabled)
4. Wait until `initializing = 0`

A port may be torn-down and re-initialized. Once `network_en` is cleared, port re-initialization should follow the “Port Initialization” flow.

12.3.4 No DRAM NIC Configuration Registers

12.3.4.1 General Configurations

Table 660 - NODNIC Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset		
1	0	9	8	7	6	5	4																										
no_dram_nic_revision								hardware_format						support_receive_filter		support_promisc_filter		support_promisc_multicast_filter				log_working_buffer_size			log_pkey_table_size							num_ports	00h
		log_max_ring_size						pd																								04h	
lkey																																08h	
cq_format																																0Ch	
node_guid																																10h-14h	
																																18h	
																																1Ch	
																								log_uar_page_size								20h	
																																24h-FCh	
port1_settings (See Table 662, “NODNIC Port Configuration Registers Layout,” on page 817)																																100h-27Ch	

Table 660 - NODNIC Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										280h-3FCh
port2_settings (See Table 662, “NODNIC Port Configuration Registers Layout,” on page 817)																															

Table 661 - NODNIC Configuration Registers Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	no_dram_nic_revision	Revision number of this interface. Must be 1.	RO
	23:16	hardware_format	Defines the Work Queues, WQE and CQE formats 0x0: ConnectX-3 0x1: ConnectX-4 See Section 12.11, “Hardware Formats,” on page 828	RO
	15	support_receive_filter	When set, 5 MAC filters or 4 MGID filters are supported for the port.	RO
	14	support_promisc_filter	When set, Promiscuous filter is supported.	RO
	13	support_promisc_multi-cast_filter	When set, Multicast Promiscuous filter is supported.	RO
	10:8	log_working_buffer_size	The base 2 Log of the size of the working buffer required by the device, given in kB.	RO
	7:4	log_pkey_table_size	The base 2 Log of the P_Key table size, given in number of entries. In dual ported NICs, each port has its own P_Key table of this size.	RO
	0	num_ports	0x0: single_port 0x1: dual_port	RO
04h	29:24	log_max_ring_size	The base 2 log of the maximum ring size given in bytes.	RO
	23:0	pd	TBD - not required in ConnectX-4/ConnectIB, ask Haggai	RO
08h	31:0	lkey	L_Key to use for send and receive WQEs	RO
0Ch	31:28	cqe_format	0x0: legacy_cqe 0x1: new_cqe	RO
10h-14h	64	node_guid	Device Node GUID	RO

Table 661 - NODNIC Configuration Registers Descriptions

Offset	Bits	Name	Description	Access
1Ch	31	support_db_record_rx	When set, the device supports reading Rx ring doorbell record from the host memory.	RO
	29	support_uar_tx_doorbell	When set, the device supports ringing Tx ring doorbell through the HCA BAR.	RO
	28	support_bar_dma_en	When set, the device supports modifying <i>dma_en</i> through the HCA BAR.	RO
	27	support_bar_cq_ctrl	When set, the device supports arming the CQ and clearing the interrupt through the HCA BAR.	RO
	26	vlan_stripping_supported	When set, NODNIC supports receive VLAN stripping. To enable VLAN stripping use the <i>vlan_stripping_en</i> bit in the port registers.	RO
20h	7:0	log_uar_page_size	Log2 (base 2) of the UAR page size, given in bytes.	
100h-27Ch	3072	port1_settings (See Table 662, “NODNIC Port Configuration Registers Layout,” on page 817)	Port 1 settings	
280h-3FCh	3072	port2_settings (See Table 662, “NODNIC Port Configuration Registers Layout,” on page 817)	Port 2 settings	

12.3.4.2 Port Configurations

Table 662 - NODNIC Port Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset																									
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2													00h																								
event (See Table 674, “NODNIC Event Word Layout,” on page 827)																																																								
																												receive_fil- ter_en	04h																											
																																																								08h
																																																								0Ch

Table 662 - NODNIC Port Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2			0											
cq_dbr_addr																												110h-114h					
								cq_n																				118h					
																												11Ch-17Ch					

Table 663 - NODNIC Port Configuration Registers Description

Offset	Bits	Name	Description	Access
00h	31:0	event (See Table 674, “NOD-NIC Event Word Layout,” on page 827)	Event word.	RO
04h	31	network_en	Enable/Disable networking through this port.	RW
	30	dma_en	Enable/Disable DMA access (without device reset). See Section 12.8, “Enabling/Disabling DMA,” on page 827	RW
	29	promisc_en	Receive promiscuous mode. When set, <i>receive_filters</i> are ignored, and no Ethernet traffic is filtered by the NIC.	RW
	28	promisc_multicast_en	Receive Multicast promiscuous mode. When set, no Ethernet Multicast traffic is filtered by the NIC.	RW
	27	vlan_stripping_en	When set, the VLAN header of a received packet will be stripped and placed the CQE.	RW
	4:0	receive_filter_en	Bitmask enabling the <i>receive_filters</i> .	RW
08h	15:0	mac[47:32]	Ethernet MAC address. Valid for both Ethernet and InfiniBand link types	RO
0Ch	31:0	mac[31:0]		

Table 663 - NODNIC Port Configuration Registers Description

Offset	Bits	Name	Description	Access
10h-4Ch	512	receive_filters_or_mgid_or_mac	Allowed received addresses. Disallowed traffic will be filtered-out. If port type is InfiniBand - mgid_filters (See Table 666, “NODNIC MGID Filters Layout,” on page 823) If port type is Ethernet - mac_filters (See Table 668, “NODNIC MAC Filters Layout,” on page 824)	
50h-5Ch	128	gid	Global IDentifier (GID).	RO
60h	31	dma_en_set	Write 0x1 to set <i>dma_en</i> . See Section 12.8, “Enabling/Disabling DMA,” on page 827	WO
	30	dma_en_clr	Write 0x1 to clear <i>dma_en</i> . See Section 12.8, “Enabling/Disabling DMA,” on page 827	WO
	15:0	lid	Local IDentifier (LID).	RO
64h	19:16	sm_sl	Subnet Manager (SM) Service Level (SL).	RO
	15:0	sm_lid	SM LID.	RO
68h	31:0	completion_address[63:32]	Pointer to the beginning of the Completion Queue.	RW
6Ch	31:12	completion_address[31:12]		
	5:0	log_cq_size	The base 2 Log of the size of the CQ given in CQ entries. Should be read after the <i>log_size</i> fields in all the rings have been set.	RO
70	31:0	working_buffer_address[63:32]	Pointer to the beginning of the Working Buffer.	RW
74h	31:12	working_buffer_address[31:12]		
78h	31:0	arm_cq (See Table 672, “NODNIC CQ Arming Word Layout,” on page 826)	CQ Doorbell register.	RW
7Ch	31:16	pkey_index	An index to the P_Key table.	RW
	15:0	pkey	Value of the P_Key in <i>pkey_index</i> of P_Key table.	RO

Table 663 - NODNIC Port Configuration Registers Description

Offset	Bits	Name	Description	Access
80h-9Ch	256	send_ring0 (See Table 664, “NOD-NIC Ring Configuration Registers Layout,” on page 821)	Transmit ring 1 parameters. For Ethernet port, this ring is the transmit ring. For InfiniBand, this ring is used for QP1.	
A0h-BCh	256	send_ring1 (See Table 664, “NOD-NIC Ring Configuration Registers Layout,” on page 821)	Transmit ring 2 parameters. For Ethernet port, this ring is reserved. For InfiniBand, this ring is a part of a general purpose UD QP	
C0h-DCh	256	receive_ring0 (See Table 664, “NOD-NIC Ring Configuration Registers Layout,” on page 821)	Receive ring 1 parameters. For Ethernet port, this ring is the transmit ring. For InfiniBand, this ring is used for QP1	
E0h-FCh	256	receive_ring1 (See Table 664, “NOD-NIC Ring Configuration Registers Layout,” on page 821)	Receive ring 2 parameters. For Ethernet port, this ring is reserved. For InfiniBand, this ring is a part of a general purpose UD QP	
108h	31:0	send_ring0_uar_index	UAR index used for send ring 0.	RO
10Ch	31:0	send_ring1_uar_index	UAR index used for send ring 1.	RO
110h-114h	64	cq_dbr_addr	CQ DB Record physical address. (cq_dbr_addr[1:0] must be 0). When updating this field, cq_dbr_addr[31:0] should be written before cq_dbr_addr[63:32]	RW
118h	23:0	cq_n	CQ number used for arming the CQ through the BAR.	RO

12.3.4.3 Ring Configurations

Table 664 - NODNIC Ring Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										00h
queue_address[63:32]																															
queue_address[31:12]																										log_size		04h			

Table 664 - NODNIC Ring Configuration Registers Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2											
doorbell (See Table 670, “NODNIC Ring Doorbell Layout,” on page 825)																												08h		
								queue_number																				0Ch		
q_key																												10h		
														pkey_index														14h		
db_record_addr_h																												18h		
db_record_addr_l																												1Ch		

Table 665 - NODNIC Ring Configuration Registers Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	queue_address[63:32]	Pointer to the beginning of the queue.	RW
04h	31:12	queue_address[31:12]		
	5:0	log_size	The base 2 log of the size of the queue given in bytes.	RW
08h	31:0	doorbell (See Table 670, “NODNIC Ring Doorbell Layout,” on page 825)		RW
0Ch	23:0	queue_number	Queue handle for this ring. Valid only after the port is enabled.	RO
10h	31:0	q_key	Infiniband Q_Key.	RW
14h	15:0	pkey_index	Index in the P_Key table.	RW
18h	31:0	db_record_addr_h	Ring DB Record physical address. When updating this field, db_record_addr_l should be written before db_record_addr_h. See Section 8.9.2, “Doorbell Record,” on page 308	RW
1Ch	31:2	db_record_addr_l		

12.4 Setting Receive Filters

The receive filters define which packets are filtered-out before they are sent to the NODNIC driver.

12.4.1 Promiscuous Mode

In this mode, no packets are filtered-out by the device. The support for this mode is indicated by *support_promisc_filter* field. The mode is set for a port by setting *promisc_en* field.

When this mode is active, other filters should not be used.

12.4.2 Promiscuous Multicast Mode

In this mode, no Multicast packets are filtered-out by the device. The support for this mode is indicated by *support_multicast_promisc_filter* field. The mode is set for a port by setting *promisc_multicast_en* field.

When this mode is active, other MAC filters should not be used for Multicast addresses.

12.4.3 Address Filters

The device supports exact address filters. Packets that do not match any of the filters are filtered-out. The support for the address filters is indicated by *support_receive_filter* field. Each address filter is set individually and activated by setting the appropriate bit in *receive_filter_en* field.

Addresses may not be modified while the address filter is activated.

For InfiniBand ports, the address filters refer to Multicast GIDs. Traffic matching these filters will be directed to the QP represented by ring1. Traffic matching the GID set by the SM is not filtered out, in addition to traffic matching the Address Filters.

Table 666 - NODNIC MGID Filters Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2													00h-0Ch
mgid_filter0																												10h-1Ch				
mgid_filter1																												20h-2Ch				
mgid_filter2																												30h-3Ch				
mgid_filter3																																

Table 669 - NODNIC MAC Filters Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	mac_filter0 (See Table 33, “MAC Address Layout,” on page 293)	MAC value matched to packet destination MAC address for filtering purposes.	RW
08h-0Ch	64	mac_filter1 (See Table 33, “MAC Address Layout,” on page 293)		
10h-14h	64	mac_filter2 (See Table 33, “MAC Address Layout,” on page 293)		
18h-1Ch	64	mac_filter3 (See Table 33, “MAC Address Layout,” on page 293)		
20h-24h	64	mac_filter4 (See Table 33, “MAC Address Layout,” on page 293)		

12.5 Sending Packets

Packet transmission flow follows the same guidelines as described in “[Posting a Work Request to Work Queue](#)”.

WQE should be written to the next empty WQBB (64 Byte slot) in the transmit ring.

Doorbell ringing by NODNIC driver is done through writing to the *doorbell* register in [Table 664, “NODNIC Ring Configuration Registers Layout”](#). If the driver wishes to use Tx doorbells through UAR (see “[Port Initialization](#)” on page 814), The index of the ring UAR is indicated in *send_ring0_uar_index*. For more details about UARs, see “[User Access Region \(UAR\)](#)” on page 249.

Table 670 - NODNIC Ring Doorbell Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											00h
								ring_pi																								

12.8 Enabling/Disabling DMA

While networking is enabled for a port, DMA may be dynamically disabled and enabled. DMA state is modified by writing to the ***dma_en*** field.

DMA can also be controlled through the HCA BAR. Setting and clearing can be done by writing 0x1 to ***dma_en_set*** or ***dma_en_clr*** respectively. Such access may not complete immediately, and the current status of DMA is indicated by the ***dma_en*** field, available through the BAR as well. The support for DMA control through the HCA BAR is indicated by ***support_bar_dam_en*** field in Table 660, “NODNIC Configuration Registers Layout,” on page 815,

When disabling DMA, driver should first wait for all transmit WQEs to be completed.

12.9 Event Handling

The driver should poll the “[Health Buffer](#)” to monitor device internal errors and other critical errors. The driver should also poll for asynchronous events by reading the ***event*** register.

The ***driver_reset_needed*** field tells the driver whether an event that requires restart with new parameters has occurred.

The ***port_management_change_event*** field notifies the driver about IB port changes. After this event, the driver needs to re-read the P_Key table, GID, and SM parameters, and re-register multicast memberships.

Table 674 - NODNIC Event Word Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0													
port_management_change_event driver_reset_needed																										link_type				port_state				00h

Table 675 - NODNIC Event Word Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	driver_reset_needed	An event requiring the driver to restart has occurred. Cleared on read.	RO
	30	port_management_change_event	Raised when one of these events occur: - LID change - P_Key table change - GUID change - client re-register Cleared on read.	RO
	4	link_type	0x0: Infiniband 0x1: Ethernet	RO
	3:0	port_state	Current port state 0x0: Down - (includes failed links) 0x1: Initialize 0x2: Armed 0x3: Active Other values are reserved	RO

12.10 Working with Other Interfaces

The option ROM drivers will need to use management commands while using the no-DRAM NIC interface. Such drivers will use icmd (TBD).

The full driver command interfaces cannot be used while in NODNIC mode. Therefore, the *cmd-q_phy_address* must have a zero value when *nic_interface* indicates NODNIC mode, and command doorbell vector must not be written. [Internal] The firmware should ignore such writes.

12.11 Hardware Formats

The Work Queues, WQE and CQE formats used by the NODNIC driver are derived from the structures defined in the Programmer's Reference Manual (PRM).

The driver should use the formats according to the *hardware_format* field. Value 0x1 indicates the formats defined by Table 660, "NODNIC Configuration Registers Layout," on page 815.

Table 676 - NODNIC Hardware Format Summary

Resource	Format
Infiniband Send Queue	UD QP. Maximum single Gather Entry.
Infiniband Receive Queue	UD QP. Maximum single Scatter Entry.

Table 676 - NODNIC Hardware Format Summary

Resource	Format
Ethernet Send Queue	Raw Ethernet QP. Maximum single Gather Entry.
Ethernet Receive Queue	Raw Ethernet QP. Maximum single Scatter Entry.
Completion Queue	64B CQE format. 64B CQE stride. VLAN stripping disabled.
WQE	See “Send WQE Format” and “Receive WQE Format”
CQE	According to <i>cqe_format</i> . See “CQE Format” and “Completion With Error”.

12.12 Resiliency and Error Recovery

12.12.1 Completion Errors

Completion errors are reported on the completion queue the same way they are reported with a full driver. The driver should always allow DMA while there is an inflight transmit request, so the firmware is allowed to write a transmit error completion. When disabling DMA, there should be no packet received so there will not be any receive completion errors. Following a completion with error, the interface must be reset in order to continue operation by performing tear down and initialization again.

12.12.2 Configuration Errors

Configuration operations through the VSEC Gateway may fail. In such an event, the syndrome bit in the Gateway will be set when the transaction is over. In addition, the data field of the Gateway will hold additional debug information. The driver should log this information for future debug, and may initiate a recovery flow (outside the scope of this document).

12.12.3 Async Errors

Asynchronous errors (both affiliated and unaffiliated) are reported through the health buffer in the initialization segment. The driver should monitor the health/catastrophe error buffers to detect device errors. Upon an error, the driver may tear down and re-initialize the device in order to proceed.

In addition, The driver should monitor the `nic_interface` field. If in any time time after driver initialization flow the value is no longer 2 (no-DRAM NIC), The driver should initiate a recovery flow (outside the scope of this document).

Part 2: Advanced Mellanox Adapter Features

13 Enhancements Summary

This section presents an overview of new features introduced in the Mellanox Adapters architecture family. It contains the following sections:

- [Section 13.1, “I/O Services’ Management and Quality of Service,” on page 831](#)
- [Section 13.2, “Network/Transport Enhancements,” on page 832](#)
- [Section 13.3, “Network Application Services,” on page 833](#)
- [Section 13.4, “Data Integrity,” on page 836](#)
- [Section 13.5, “Memory Management,” on page 836](#)

13.1 I/O Services’ Management and Quality of Service

13.1.1 Virtualization Support

The device extends virtualization support of its predecessor by adding full HW support for virtual network ports.

On host side the device fully supports PCI SR-IOV, supporting up to 256 virtual functions.

The device integrates an IOMMU cache. Graceful page faults are supported on IOMMU address translation path, thereby enabling the Hypervisor to oversubscribe I/O memory in VM environment. If I/O operation experiences an IOMMU translation error, the operation is halted and an event is generated to the Hypervisor. After the Hypervisor signals the HW that memory mapping is resolved, the device resumes I/O operation.

The device extends virtualization support for InfiniBand by integrating a virtual InfiniBand switch, enabling each virtual function to be exposed as a fully capable HCA device, including its own QP0 and QP1. This allows each virtual machine to run the full InfiniBand software stack.

Virtualization support is discussed in detail in [“Virtualization” on page 838](#).

13.1.2 Send QoS

The device supports precise provisioning of I/O services (QoS) on send path at very low granularity. QoS can be enforced on packet granularity and high-priority I/O channels are guaranteed to be served at high priority even in presence of high operational load on other channels.

13.1.3 Fast Path for Latency Sensitive Traffic

The device supports providing fast service for latency sensitive traffic on both send and receive path. Latency sensitive QPs are marked with *latency_sensitive* bit in the QP context.

13.2 Network/Transport Enhancements

13.2.1 Dynamically Connected Transport Service

Dynamically Connected transport (DCT) service is an extension to transport services to enable a higher degree of scalability while maintaining high performance for sparse traffic. Utilization of DCT reduces the total number of QPs required systemwide by having RC type QPs dynamically connect and disconnect from any remote node. DCT connections only stay connected while they are active. This results in smaller memory footprint, less overhead to set connections and higher on-chip cache utilization and hence increased performance. This transport service is discussed in details in [“Dynamically Connected Transport Service” on page 862](#).

13.2.2 Enhanced Completion Reporting

13.2.2.1 Extended CQE

The device extends CQE to optionally carry the following additional information:

- I/O operations' execution (timestamp)
- small Atomic and read responses
- data received for small receive operations
- GRH for UD responder

Utilizing extended CQEs can reduce the number of memory reads and writes during an I/O operation as well as save CPU/chipset cache misses. For details, see [“CQE Format” on page 435](#).

13.2.3 Dual Write

The device enables the posting of a single RDMA write Work Request on a send Work Queue and have this request executed on two different Send Queues, one that is used when posting the request and another one with an adjacent QP number. Such an operation will result in two messages sent by the device. Each one will have a distinctly different address in the RDMA header and will send the same data to two destinations. The data will be read once on PCI and sent twice on the network. Completions will be reported separately for each of the two Send Queues. Use of the dual write feature may significantly reduce PCI bandwidth use.

13.2.4 Received MADs Processing De-multiplexing

The device provides hooks to demultiplex received MADs to either be processed by firmware internally or be forwarded to the host CPU. This de-multiplexing is done based on Class/Attribute/Method of the incoming MAD. [The configuration of this mechanism is done via the INIT_MAD_DEMUX command.](#)

13.2.5 Exponential Backoff Timer

[Internal] This feature is marked internal until confirmed and implemented.

Infiniband Reliable Connection utilizes timeouts and retransmissions to handle packet loss. Choosing the right value for the Local ACK Timeout is challenging. If too small a value is cho-

sen, the number of packet transmission timeouts reported by the remote CA may increase, which may increase the amount of work required in the CA to successfully send a packet. In addition, in most situations involving congestion and busy devices, precipitate retries are not helpful and may create or add to congestion. If too large a value is chosen, the amount of time that it takes to notice that a packet has not been successfully transmitted (e.g. due to a CRC error on the wire) will be increased, which may increase the amount of time it takes to recover from or report such errors.

The device implements, per QP, an algorithm for increasing back-off interval each time it encounters a situation where it needs to retransmit the same message. An operation is to be retried, after the sender waited the timeout based on Local ACK timeout. For every failed attempt the timeout value is multiplied by a Retry Timeout Multiplier (RTM) value. When retransmission succeeds, the Local ACK Timeout is restored to its original value.

13.2.5.1 Configuration

The support for activating an Exponential Backoff timer is indicated by *exponential_backoff* field in “QUERY_HCA_CAP – Query Device Capabilities”.

The RTM value is set in the *log_rtm* field in Table 21, "Address Path Structure Layout", where value 0x0 renders the feature disabled (Infiniband Specification compliant). The value is set and modified in conjunction with the Local ACK timeout (*ack_timeout* field) using QP state transition commands “INIT2RTR_QP – Init to RTR QP”, “RTR2RTS_QP –RTR to RTS QP” and “RTS2RTS_QP – RTS to RTS QP”

Note: The timeout value saturates at the maximal value allowed for *ack_timeout* field.

13.3 Network Application Services

13.3.1 Precise Clock Synchronization Over the Network

Some applications require support in precise clock. IEEE 1588 standard provide a mechanism for accurate synchronization between the distributed entities over the network

The device implements HW hooks enabling implementation of IEEE-1588 version 2 network clock synchronization protocol. The device HW supports clock synchronization across network within sub-microsecond accuracy.

Optionally, completion reports can include precise timestamp when the I/O operation generating this completion is executed, hereby enabling the application to know when I/O operation (send or receive) was actually executed by the device.

13.3.1.1 Auxiliary Pulse-Per-Second (PPS) Signals

The device supports GPIO pin(s) which can be used for PPS in or PPS out signals.

Both pps out and pps in signals use the core clock.

The PPS functionality is using two registers:

- ***MTPPS*** – Used to configure and query PPS parameters. The configuration is supported only when `HCA_CAP.pps_modify == 1`, and the query supported only when `HCA_CAP.pps == 1`. See TBD
- ***MTPPSE*** – Used to configure the PPS event generation mode. Supported only when `HCA_CAP.pps == 1`.

13.3.1.1.1 Selecting PPS pins functionality:

The device pins used for PPS signals are configurable GPIO pins. The user shall verify the device capabilities through ***MTPPS.cap_number_of_pps_pins*** which retrieve the total number of pins which supports 1pps signals. In addition user shall verify the maximum number of pps in and pps out pins through ***MTPPS.cap_max_num_of_pps_in_pins*** & ***MTPPS.cap_max_num_of_pps_out_pins***.

The applicable configuration for each pin is retrieved through ***MTPPS.cap_pin_x_mode***. Each pin can support pps in, pps out, or both pps in. The user shall configure the exact functionality per pin via ***MTPPS.pin_mode***.

13.3.1.1.2 Pulse-Per-Second (PPS) Output:

The device can indicate, through a GPIO pin, when the internal clock exceeds a value. This value is set by SW through the command interface.

Enablement of generation of PPS output is done through the ***MTPPS.enable***.

The device support modes of output PPS signal:

- ‘one shot’ signal – single pulse
 - The ***MTPPS.pattern*** shall be set to ‘pulse’ pattern.
 - ***MTPPS.out_pulse_duration*** shall be set (or use default)
 - The software shall set next pulse time in ***MTPPS.time***.
 - On the raising edge of the pulse, an event can be generate. To enable receiving an event:
 - Register to the PPS event. See Section 8.20.12.6, “PPS Event - Pulse Per Second Event,” on page 494
 - Enable and arm the event using MTPPSE register by setting `event_arm` and `event_gen_mode` fields.
 - After each event, SW must make sure to re-arm the event mechanism by setting `event_arm` in MTPPSE.
- ‘periodic’ signal – pulse every 1 second according to device clock
 - The ***MTPPS.pattern*** shall be set to ‘periodic’ pattern
 - ***MTPPS.out_pulse_duration*** shall be set (or use default)
 - Set 1st pulse time in ***MTPPS.time***
 - The pulse will be generated every 1 second according to device clock.

- A correction to device clock can be entered through *MTPPS.out_periodic_adjustment*. 'Next signal time' is set to 'previous signal time' $\pm$ repetitive_offset.
- Events behavior is the same as in 'one shot' mode.

13.3.1.1.3 Pulse-Per-Second (PPS) Input:

The device has an input for electrical signals that indicates the start of a second. This input can come from a precision clock generator such as GPS. The device can synchronize its internal clock to this signal.

Enablement of receiving of PPS signal is done through the *MTPPS.enable*.

The *MTPPS.pattern* shall be set to rising or falling edge pattern.

Every time a signal is sensed (according to the defined pattern) the time will be latched in MPGO.in_time and *MTPPS.time*. In addition event will be set through ptp event register (NIC only)

The PTP events are cleared on read. If the event latched time has not been read and another event was received, the new event timing will be stored instead of the old latched timing. In that case, a new (additional) event will be generated.

Events behavior is the same as in PPS out 'one shot' mode.

13.3.1.2 Congestion Control Improvements

Golan supports congestion control by DLID.

13.3.2 Extended Atomic Operations

The device further extends the atomic operations implemented on its predecessor allowing larger operands to be used for single instruction multiple data (SIMD) atomic operations. The device supports masked *fetch&add* and masked *compare&swap* SIMD operations are in addition to standard InfiniBand atomic operations. Operands' sizes for SIMD operation range from 1 to 256 bytes; operands' size must be a power of two bytes.

The device also supports atomicity between RDMA WRITE operations and atomic operations. This is to allow semaphore releasing by a write operation. This ability should be enabled explicitly per QP (QP/DCT.atomic_like_write_en).

13.3.3 Programmable Offload Engines

Golan integrates a programmable microprocessor that can be used to execute network operations. In a sense, a process or thread from the host can be offloaded to the HCA. HCA execution resources (work queues, completion queues and event queues) can be associated with this integrated processor and program running on this processor can handle network operations as well as accessing host memory. This capability will be used to exploit user thread offload to the HCA. More details can be provided upon request.

13.3.4 Programmable Offload Engines

Golan integrates a programmable microprocessor that can be used to execute network operations. In a sense, a process or thread from the host can be offloaded to the HCA. HCA execution resources (work queues, completion queues and event queues) can be associated with this integrated processor and program running on this processor can handle network operations as well as accessing host memory. This capability will be used to exploit user thread offload to the HCA. More details can be provided upon request. CRQ is marked by the *crq* bit in the QP context.

13.3.5 Contiguous SRQ

Unlike regular SRQ, where the WQEs are not contiguous and there is a link list connecting them with the contiguous SRQ, WQEs must be contiguous. In this mode, SW must post WQEs to SRQ in contiguous manner and HW will use this to do performance optimizations.

13.4 Data Integrity

13.4.1 Internal (HW) Data Protection

The device provides data integrity check and correction on its external interfaces, internal arrays, and internal logic. The device provides full end to end data path protection. An integrity check failure will cause the chip to reset immediately. A log for the cause of this reset will keep its value over this reset.

For more details, please refer to [“Hardware-level Data Integrity” on page 877](#).

13.4.2 Control Structures’ Integrity Checks

As an additional protection, the device implements consistency checks on all host memory resident control objects to avoid bogus execution in chimerical event of software error such as memory-resident control structures being corrupted by bogus SW. For more details, please refer to [“Software-level Data Integrity - Control Objects’ Consistency Checks” on page 878](#).

13.4.3 Application Level Data Integrity Verification and Signature Handover

The device supports application level data integrity checks via a signature handover mechanism. Various signature types, including XOR8, CRC32, and DIF can be automatically calculated and checked, stripped or appended during the transfer at full wire speed.

This capability is detailed in [“Application Data Integrity: Signature Handover Operations” on page 880](#).

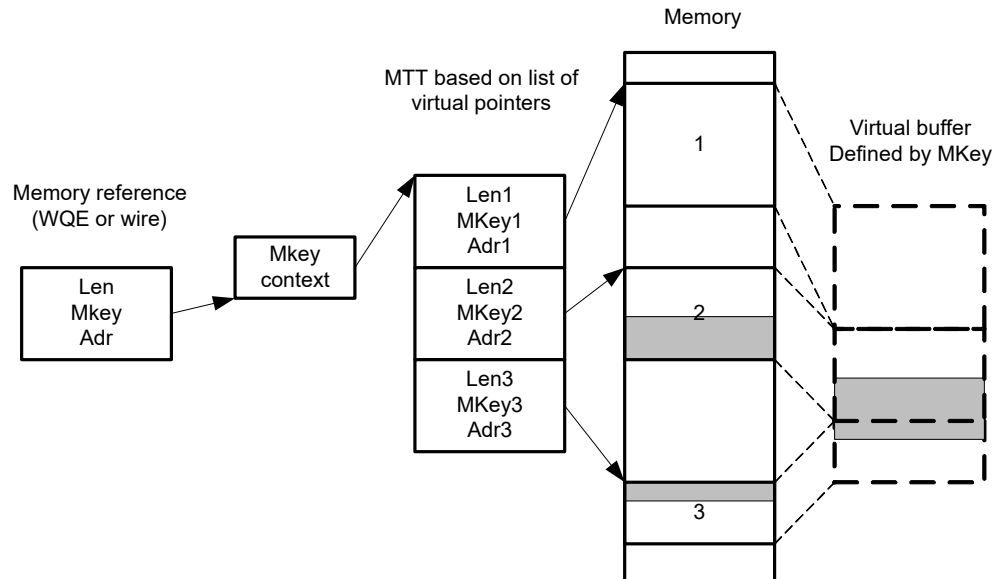
13.5 Memory Management

13.5.1 User Memory Registration

The device supports user-level memory registration - converting multiple chunks of pre-registered memory to a single virtually-contiguous address space that can subsequently be used as a single virtually-contiguous memory buffer. Such a mechanism is implemented by introducing a

new WR that supports an unrestricted scatter-gather list implementing an additional level of indirection of address translation as shown in [Figure 53](#).

Figure 53: Address Translation Indirection



This capability enables having a scatter-gather list as a target for RDMA operation as well as other related benefits. In particular, it provides fast memory registration (previously known as FMR/FRWR) capabilities to user-mode and does not require software to mess around with cache flushes or ICM exposure. See [“User-Mode Memory Registration \(UMR\)” on page 918](#) for details.

13.5.2 Enforce Ordering on RDMA Read Responses

When multiple RDMA read requests are sent from a requester to a responder, the device has the ability to fence between the PCIe read requests used by the responder to fetch data from host memory. This property is configured in the responder's MKey.

13.5.3 Enforce Ordering on Gathering Data to be Sent

When multiple WQEs are executed on an SQ, the device will usually initiate read requests to gather the data for multiple WQEs in parallel. The purpose of this is to maximize bandwidth usage on the PCI link between handling of two WQEs.

The device also has the ability to enable fencing between the PCIe read request of different WQEs. This is configured per WQE by setting the fence-mode field in the WQE.

13.5.4 HCA Control Objects' Management

The device reduces software overhead associated with management of HCA control memory (ICM) by managing ICM memory on its own. Hence, the software interface is greatly simplified from its predecessor.

14 Virtualization

14.1 Ethernet Virtualization

The device supports state-of-the-art virtualization technology on Ethernet networks. Device resources are made available to virtual machines (VMs) in an efficient and secure way, so one virtual machine does not affect the normal operation of another VM.

The device supports partition of the PCI to multiple Physical Functions (PFs) in hierarchical model as shown in the later subsections in details. Each PF can be connected to a single Network Port as below:

- PF0 and PF2 mapped to Physical Port 1.
- PF1 and PF3 mapped to Physical Port 2.

The device also supports PCI SR-IOV, enabling to partition a PF to logical devices - Virtual Functions (VFs) - and assign these VFs to different virtual machines. A VM is assigned a vNIC or multiple vNICs. **The device also supports a 2nd PCI BAR, enabling to partition a PF/VF to multiple Sub- Functions (SFs). Each SF provides a separate vNIC namespace and has its own vport (which provides a separate connectivity to the network). See Section 14.3, “Sub-Function,” on page 845.**

The device enables to implement a single-driver model where the same driver (binary) can be used in native OS environment, Virtual Machine Monitor (VMM) and Virtual Machine (guest VM).

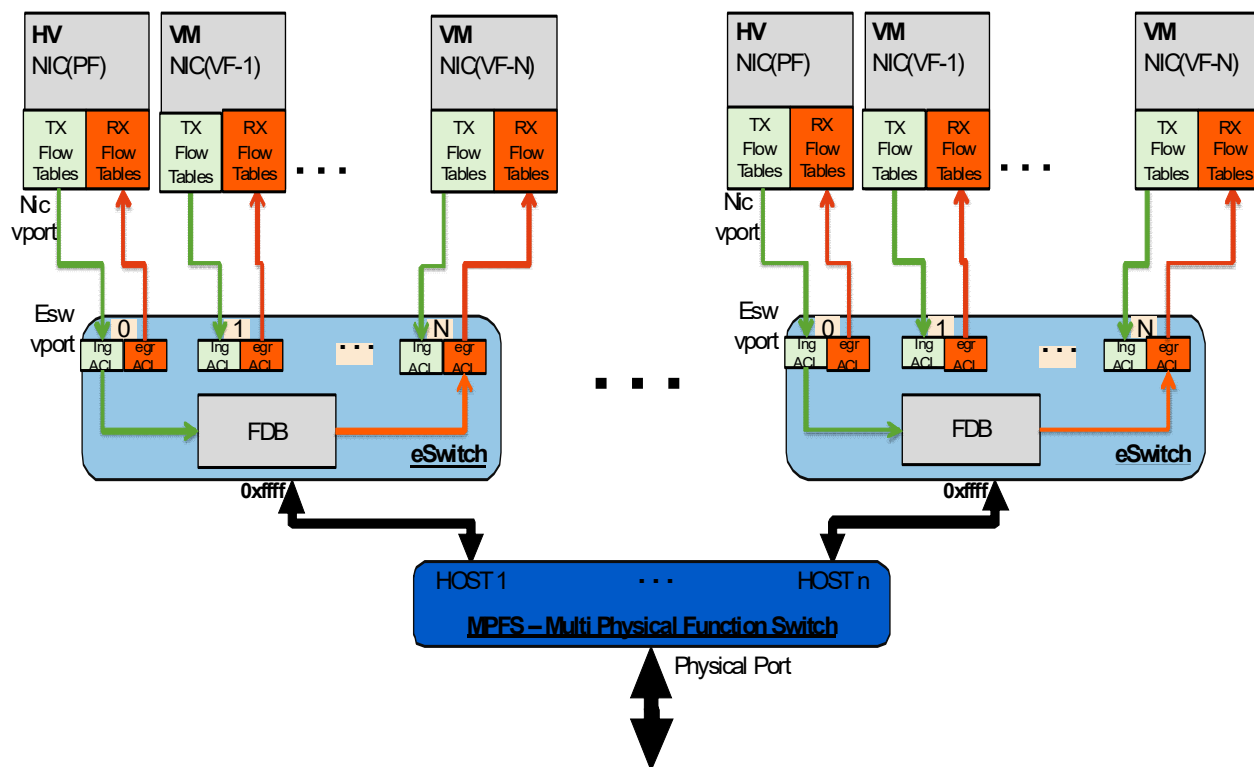
14.1.1 Components of Ethernet Virtualization

Ethernet virtualization requires the following components:

- vNICs - a vNIC is a virtual network interface card which provides almost all physical NIC functionalities. A VM is assigned a vNIC or multiple vNICs. Each vNIC interface is connected to the network through a PF/VF which in turn is connected to a virtual port (vport) of an e-switch. See [Figure 54](#).
- E-switch - An e-switch connects the vNICs to the device through its vports. The e-switch's main responsibility is to manage the traffic between VM to VM and VM to external network and other switch functionalities such as ACLs, Encap/Decap, counters and so on. For further details, see [Section 14.1.3, “E-switch,” on page 841](#).
- Virtual Ports - The virtual ports (vports) connect the vNICs (nic_vport) to the device through the e-switch's vport (esw_vport). For further details, see [Section 14.1.2, “Virtual Ports,” on page 839](#).
- UpLink - Each e-switch is connected to the external network by one UpLink which is managed by the e-switch manager, and passes through a Multi-Physical Function Switch (MPFS) if more than one PF uses the same physical port (e.g. in cases where multiple hosts use the device). For further details, see [Section 14.1.4, “UpLink,” on page 843](#).

Figure 54 illustrates a high level model for Ethernet virtualization. This model can also illustrate a non-virtualized device as a special case.

Figure 54: Ethernet Virtualization Model



14.1.2 Virtual Ports

Each vNIC can access the external network through its VF. Each vNIC is connected to the device through a `nic_vport` which in turn is connected to an e-switch by an `esw_vport`.

Each one of the vports (`nic_vport` or `esw_vport`) has its own state (see [Section 14.1.2.1](#)) and context (see [Section 14.1.2.2](#)) which is controlled and managed using the Vport commands listed in [Section 29.31, "Vport Commands,"](#) on page 2154.

Each physical function (PF) always connects to its own e-switch via `esw_vport zero` and manages its e-switch by `HCA_CAP.vport_group_manager==1`. The vNICs VFs of each PF must be connected to the same `esw_vports` of their PF and in the same order as their functions numbering. That is, VF1 should be connected to `esw_vport_num=1`, VF2 to `esw_vport_num=2`, VF n to `esw_vport_num=n`.

Each vNIC has the responsibility to query and modify state, addresses, filters and other parameters of its own `nic_vport` (depending on the permission in context - for example, permanent mac-address cannot be modified). `esw_vport` state and parameters can be queried and modified only by the e-switch manager - `HCA_CAP.eswitch_manager` (See [Table 1571, "HCA Capabilities Layout,"](#) on page 1671).

vNIC which access its own nic_vport does not know its own vport number (by setting other_vport=0 in the used vport command). See [Section 29.31, “Vport Commands,” on page 2154](#).

The e-switch manager which is allowed to access the esw_vports and nic_vports of VFs connected to the e-switch it manages, should specify the vport_number and type (nic_vport or e-switch vport) that it is trying to access. In this case, the vport_number used is equal to the VF order in the list of VFs of a specific PF. That is, first VF becomes connected to vport_number=1, second VF to vport_number=2 and so on.

14.1.2.1 Vport State

After an e-switch manager performs INIT_HCA command, its e-switch is created and all its esw_vports gets a DOWN state except of the esw_vport of the eswitch manager. In addition to this, the nic_vport of this PF will get an UP state.

After a VF performs INIT_HCA command, its nic_vport state changes from DOWN state to UP state. Each function (PF or VF) can query and modify its nic_vport state using QUERY_VPORT_STATE and MODIFY_VPORT_STATE commands (See [Section 29.31, on page 2154](#)). E-switch manager is also allowed to change the state of each esw_vport belonging to its e-switch using the same commands.

esw_vport state can be modified to be UP, DOWN or FOLLOW (follow uplink state), and nic_vport state can be modified only by its VF to be UP or DOWN.

The state which the QUERY_VPORT_STATE command returns when querying the nic_vport state takes into account the state of the esw_vport, i.e. when return UP only if both states are in UP state.

[Table 677, “Vport State Summary”](#) summarizes all actions regarding nic_vport and esw_vport state.

Table 677 - Vport State Summary

	Controlled by	States allowed by modify_vport_state	States Returned by query_vport_state
nic_vport	Each function (PF or VF) controls only its nic_vport	DOWN/UP	DOWN/UP taking into account the esw_vport state
esw_vport	E-switch manager	DOWN/UP/FOLLOW	DOWN/UP taking into account the uplink state if configured to be follow.

In the initialization stage, software should perform QUERY_VPORT_STATE command to get its nic_vport state as explained in [Section 7.2, “HCA Driver Start-up,” on page 244](#).

Software should also register to the PORT_STATE_CHANGE event to get an indication whether port state has changed. If software retrieves this indication, this signifies that the state has been changed and a QUERY_VPORT_STATE command should be performed to get the new state.

14.1.2.2 Vport Context

Each type of vport has its own context (See [Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284](#) and [Section 8.7, “NIC_Vport Context - NIC Virtual Port Context,” on page 287](#)) which allocates all the required static and dynamic parameters and characteristics of the connection (i.e. permanent addresses, current mac addresses, VLANs list, anti-spoofing, etc).

Software can query the context using the commands `QUERY_ESW_VPORT_CONTEXT` and `QUERY_NIC_VPORT_CONTEXT`, and can also modify some of the context’s fields, if allowed, using the commands `MODIFY_ESW_VPORT_CONTEXT` and `MODIFY_NIC_VPORT_CONTEXT` (See [Section 29.31, “Vport Commands,” on page 2154](#)).

14.1.2.2.1 NIC Vport Context

In this initialization stage, each function (PF or VF) should query its own `nic_vport` context to clarify the basic information about its port, i.e. like permanent mac and allowed VLANs in VGT+ mode. See [Table 29, “NIC_Vport Context Layout,” on page 287](#).

It can also modify some of the fields depending on the permission of this field appeared in the same context.

It should also set the addresses and VLANs which are allowed to arrive to the NIC.

vport group manager (`HCA_CAP.vport_group_manager`) is allowed to manage and modify some parameters of the `nic_vports` connected to the e-switch it manages. It is responsible for example to set the permanent addresses of each VF connected to the e-switch under its responsibility. Detailed information regarding this in `MODIFY_NIC_VPORT_CONTEXT` Bitmask ([Table 35, “MODIFY_NIC_VPORT_CONTEXT Bitmask,” on page 293](#)).

14.1.2.2.2 E-switch Vport Context

Only e-switch manager (`HCA_CAP.eswitch_manager`) is allowed to query and modify the contexts of `esw_vports` belonging to the e-switch under its responsibility.

After a e-switch manager performs `INIT_HCA`, no traffic will be allowed to pass through this e-switch to any vNIC, and e-switch manager should modify the `esw_vports` state to UP/FOLLOW in order to allow traffic on this vports.

See [Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284](#).

14.1.3 E-switch

As mentioned above, each vNIC can be connected to the device through a vport of an e-switch. Each e-switch has a single manager (usually the PF), indicated by `HCA_CAP.eswitch_manager==1` (see [Table 1571, “HCA Capabilities Layout,” on page 1671](#)), which is responsible for managing and controlling the e-switch and the vNICs connected to its `esw_vports`

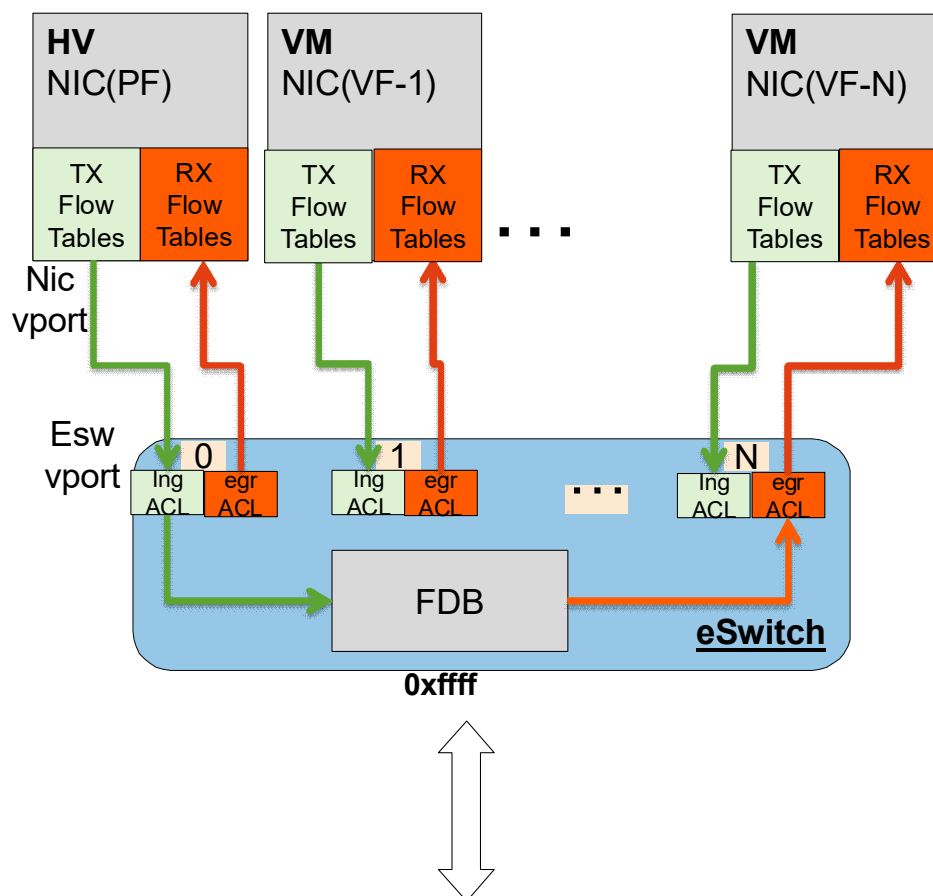
`HCA_CAP.eswitch_manager` is responsible for forwarding each packet to the right vport.

In addition to these forwarding decisions, for each `esw_vport`, Ingress and Egress ACL modules can be allocated which can perform some ACL functions like Drop, Encapsulation and Decapsulation. All these modules are managed by Flow tables components. See [Section 29.18, “Flow Table Commands,” on page 1985](#).

Note that all eswitch contexts and commands can be accessed and modified only by eswitch manager(HCA_CAP.eswitch_manager==1) (for example eswitch vports state, eswitch vport context, uplink, eswitch flow tables....).

Figure 55 describes the e-switch structure.

Figure 55: E-switch Structure



14.1.3.1 E-switch FDB

The e-switch FDB is the main module of the e-switch and it is responsible for forwarding decisions, i.e. decides to which vport (or external port) any packet should be sent to.

This module is built as a TCAM functionality, and can be configured and managed using flow table mechanisms - see [Section 29.18, “Flow Table Commands,” on page 1985](#).

Before creating the e-switch, all the received packets are transmitted to the e-switch manager.

The e-switch manager (HCA_CAP.eswitch_manager) must set the needed rules to perform the required forwarding.

Note that the e-switch FDB is also able to make drop packet decisions depending on the headers supported by it.

14.1.4 UpLink

As explained above, UpLink is the link which connects the e-switch to the physical port, and is also managed by the PF which manages the e-switch.

UpLink stays in DOWN state until the e-switch manager performs INIT_HCA, then it moves to a FOLLOW state. This state means the UpLink state follows the state of the physical port.

E-switch manager can query and modify this state using QUERY_VPORT_STATE and MODIFY_VPORT_STATE commands (see [Section 29.31, “Vport Commands,” on page 2154](#)).

It can change the state of the UpLink to UP, DOWN or FOLLOW.

Because each PF has just one Uplink, it should access it only by vport_num = 0xffff.

14.1.4.1 ACLs

Each vport of an e-switch has two ACL modules - ingress and egress - that can provide ACL functionalities like drop, count, Encap/Decap and is managed also by flow table mechanisms - see [Section 29.18, “Flow Table Commands,” on page 1985](#).

14.1.5 Virtual NIC Start-Up

After a driver performs the start-up instructions described in [Section 7.2, “HCA Driver Start-up,” on page 244](#), it should perform the following instructions:

- Query NIC_VPORT_CONTEXT to determine the permanent MAC address and other important parameters.
- Set the current_mac_address (can be more than one) using MODIFY_NIC_VPORT_CONTEXT. This address specifies the address(es) the vNIC is going to use. Note that not each vNIC has the permission to set current_mac_address which is different from the permanent mac.

Beside the above instructions, e-switch manager should perform the following:

- Set a permanent MAC address for each vNIC connected to the e-switch it manages through MODIFY_NIC_VPORT_CONTEXT.
- Create and configure the e-switch FDB module of the e-switch to ensure each vNIC gets its traffic. See [Section 14.1.3.1, “E-switch FDB,” on page 842](#).
- Configure parameters of the esw_vports using MODIFY_ESW_VPORT_CONTEXT.
- Optional - configure Ingress and Egress ACL for each esw_vport.
- Execute ENABLE_HCA for the vNIC to allow it to use the e-switch. See [Section 29.3.15, “ENABLE_HCA,” on page 1823](#).

Note: Without executing this command, vNICs which are not e-switch managers will not be able to perform start-up.

14.1.6 NIC Vport Change Notification

Note: This mechanism is supported only when `HCA_CAP.nic_vport_change_event==1`.

E-switch manager can be notified of changes in `nic_vport_context` in any vport belonging to the e-switch it manages by registering to the asynchronous event NIC Vport Change Event (See [Section 8.20.14.3, “NIC Vport Change Event,” on page 502](#)).

To get the change notification, e-switch manager flow performs the following:

- Creates EQ and registers to `Nic_Vport_Change_Event` (See [Section 8.20.14.3, “NIC Vport Change Event,” on page 502](#)).
- Sets which fields of the NIC Vport Context to generate the event by setting `event_on_uc_address_change/event_on_mc_address_change/event_on_vlan_change/event_on_promisc_change` in `nic_vport_context` (See [Table 29, “NIC_Vport Context Layout,” on page 287](#)).
- Sets `arm_change_event` in NIC Vport Context to 1.
- After a change happens, the device generates a NIC Vport Change Event (See [Section 8.20.14.3, “NIC Vport Change Event,” on page 502](#)) and sets `arm_change_event` to 0. The device does not generate another event until e-switch manager rearms by setting `arm_change_event` to 1 again. Thus, after E-switch manager gets the event, it should read the `nic_vport_context` which causes the event and set `Nic_Vport_Context.arm_change_event` to 1 again.

14.2 Resource Management for Virtual NICs (vNICs)

Note: This feature is supported capable only when `HCA_CAP.vhca_resource_manager==1`.

An e-switch manager can modify, control and manage the allowed features and the maximum resources for vNICs which belong to the e-switch it manages. This modifies part of the capabilities reported by `QUERY_HCA_CAP` (See [Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667](#)). The modifiable capabilities are indicated in the description tables of these fields.

In order to modify these capabilities, vNIC group manager (`HCA_CAP.vhca_group_manager==1`), that is capable of managing resources (indicated by `HCA_CAP.resource_manager==1`) should:

- Query the maximum capabilities of the structure that allocates the field(s) it wishes to modify by using `QUERY_HCA_CAP` command of the specific function by setting the required `function_id`.
- Modify the needed fields and only if they are modifiable as indicated in description tables of these fields.
- Set the new structure that includes the new required capabilities for a specific function by using `SET_HCA_CAP` command (See [Section 29.3.5, “SET_HCA_CAP – Set Device Capabilities,” on page 1803](#))

- As the device has its own maximum limitations on resources and features toward vNICs, it is the SW responsibility to perform QUERY_HCA_CAP again and check the new values.

14.2.1 Trusted virtual NIC

Drivers running on top of a virtual NIC, are limited to the resources and features allowed by the device and the e-switch resource manager. These drivers are not trusted, and may try to exploit vulnerabilities of the device interface. In case the virtual NIC driver is trusted, the eswitch resource manager may set `trusted_vnic_vhca` field in the virtual NIC capabilities. This configuration will relax the restrictions imposed on the virtual NIC driver.

[Internal] clearing this bit disables: modifying entity (page) size by UMR, modifying atomic privileges by UMR, creating UMR-able indirect MKeys (in case repeated KLMs were disabled as well), creating MKeys with bad entity length, creating indirect MKeys with misaligned atomics.

14.3 Sub-Function

When `.sf=1`, the device supports creating sub-functions on additional PCI BAR called BAR2.

TBD - take from Achiad's DOC.

14.4 Multi-port vHCA

When `HCA_CAP.num_ports=1` and `HCA_CAP.num_vhca_ports=1`, each PCI function drives its own (native) network port. This is the Function-Per-Port model that is commonly used by OSes networking infrastructures – each PCI function represents a single network port and the OS instantiates a single network device interface per such PCI function.

The IB spec, however, defines multi-port HCA devices where RDMA transport objects associated with different network ports can share transport objects. e.g QPs of different IB device ports can use a shared Memory Region.

In this function-per-port architecture, each PCI function has its own objects namespace. Therefore, objects of one PCI function (e.g QP) cannot be associated with objects of another PCI function (e.g CQ).

To support multi-port IB devices resource sharing, the device introduces a multi-port VHCA that allows to associate transport resources of a PCI function with any port of the device (including non-native ports).

The feature support is indicated by `HCA_CAP.num_vhca_ports>1`. To enable the feature, SW should:

- Set the current `HCA_CAP.num_vhca_ports` with the value advertised at `max HCA_CAP.num_vhca_ports`. Once a multi-port VHCA is enabled, the port field of its transport objects (QP.port, ROCE_ADDRESS_TABLE.port) denotes the port number in the global device ports namespace rather than in the function's namespace (as in the Function-Per-Port case).

- Set the affiliation - a non-native VHCA port (that is not associated with the VHCA's native port) must be explicitly affiliated by the driver to the NIC vport of the VHCA that is natively associated with it. Note that the affiliation is allowed only between VHCAs assigned with the same `sw_owner_id`.

The affiliation is enabled by setting `NIC_VPORT_CONTEXT.affiliate_criteria` and VHCA being affiliated to its specified NIC vport in the `NIC_VPORT_CONTEXT.affiliated_vhca_id` field (See [Section 14.1.2.2.1, "NIC Vport Context," on page 841](#)).

The `NIC_VPORT_CONTEXT.affiliation_criteria` field specifies the condition for routing traffic hitting the NIC vport to the affiliated VHCA. Traffic not meeting the specified criteria will be natively routed to the native VHCA of the NIC vport. When a non-native VHCA port is not affiliated with any NIC vport, traffic sent on that VHCA port (of QPs associated with it) will be silently dropped by the device (as if the VHCA port link state is down).

Note: When `HCA_CAP.num_vhca_ports > 1`, the PCI function's driver can associate only the following transport objects with a port:

QP
ROCE_ADDRESS_TABLE

Note: TISs and TIRs keep being implicitly associated with the native port of the function that has created them (per the Function-Per-Port model), as when the multi-port VHCA capability is not enabled

Note: The device's E-switch/port are considered outside the VHCA context. Thus, they keep being managed via the PCI function that natively owns them, as when the multi-port VHCA capability is not enabled

TBD - add figure.

14.5 Ethernet Virtualization in CPU Devices (BlueField)

BlueField device is a multi-core ARM processor in a highly integrated system on chip (SoC), optimized for NVMe storage systems, network processing applications, GPU and Deep Learning systems, security systems, and embedded appliances.

This section describes the virtualization model in BlueField device where ARM processors are presented and defines the relationship between the NIC, ARM processors, and the regular external host(s).

BlueField device can be presented in one out of two models:

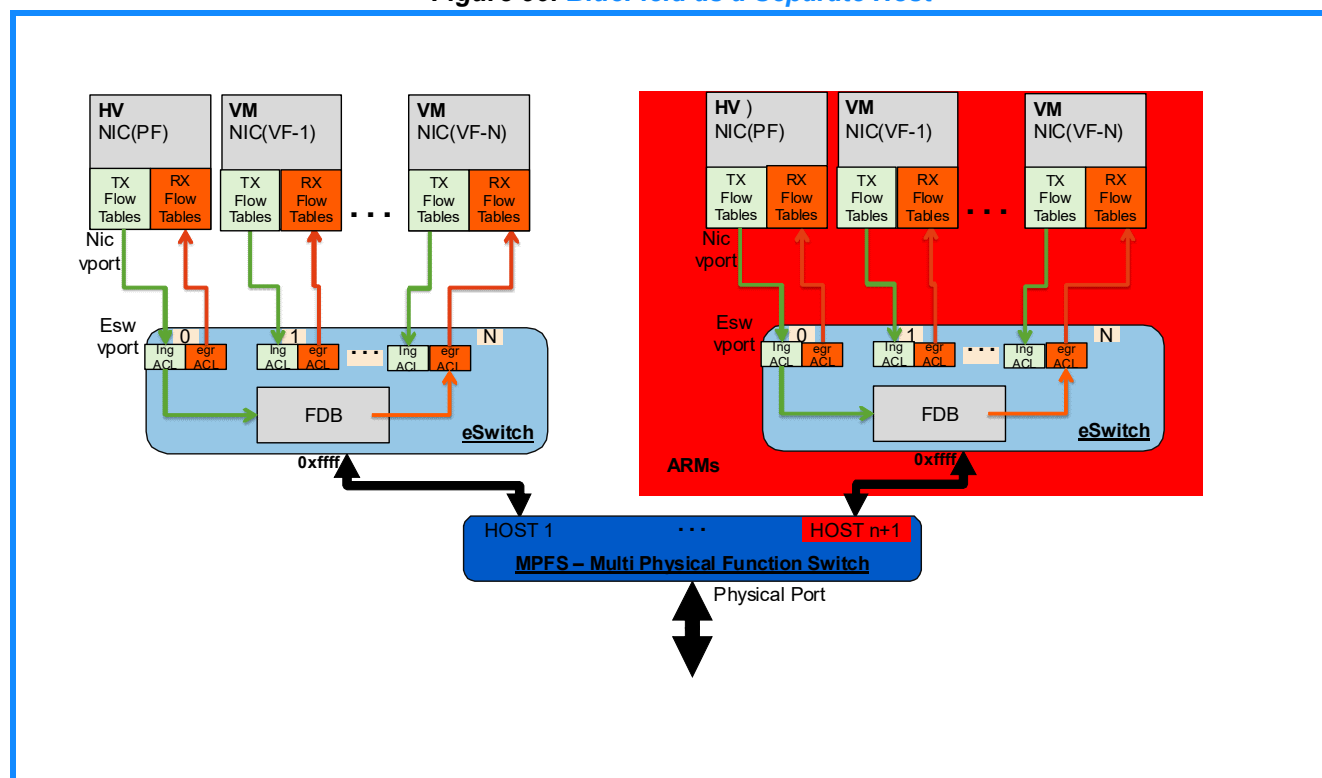
- ARM processors as a separated and orthogonal host like any other external host. See [Section 14.5.1, "Model 1 - Separated and Orthogonal Host," on page 847](#).
- ARM processors as Embedded CPU (EC) and part of the external hosts virtualization model. See [Section 14.5.2, "Model 2 - Embedded CPU \(EC\)," on page 847](#).

The BlueField virtualization model is configured by a non-volatile configuration NV_INTERNAL_CPU_CONF (See See Table 394, “NV_INTERNAL_CPU_CONF Layout,” on page 664).

14.5.1 Model 1 - Separated and Orthogonal Host

In this model, ARM processors of the BlueField device behave as a regular host like any other external host in the multi-host virtualization model (See Figure 56, “BlueField as a Separate Host,” on page 847).

Figure 56: BlueField as a Separate Host

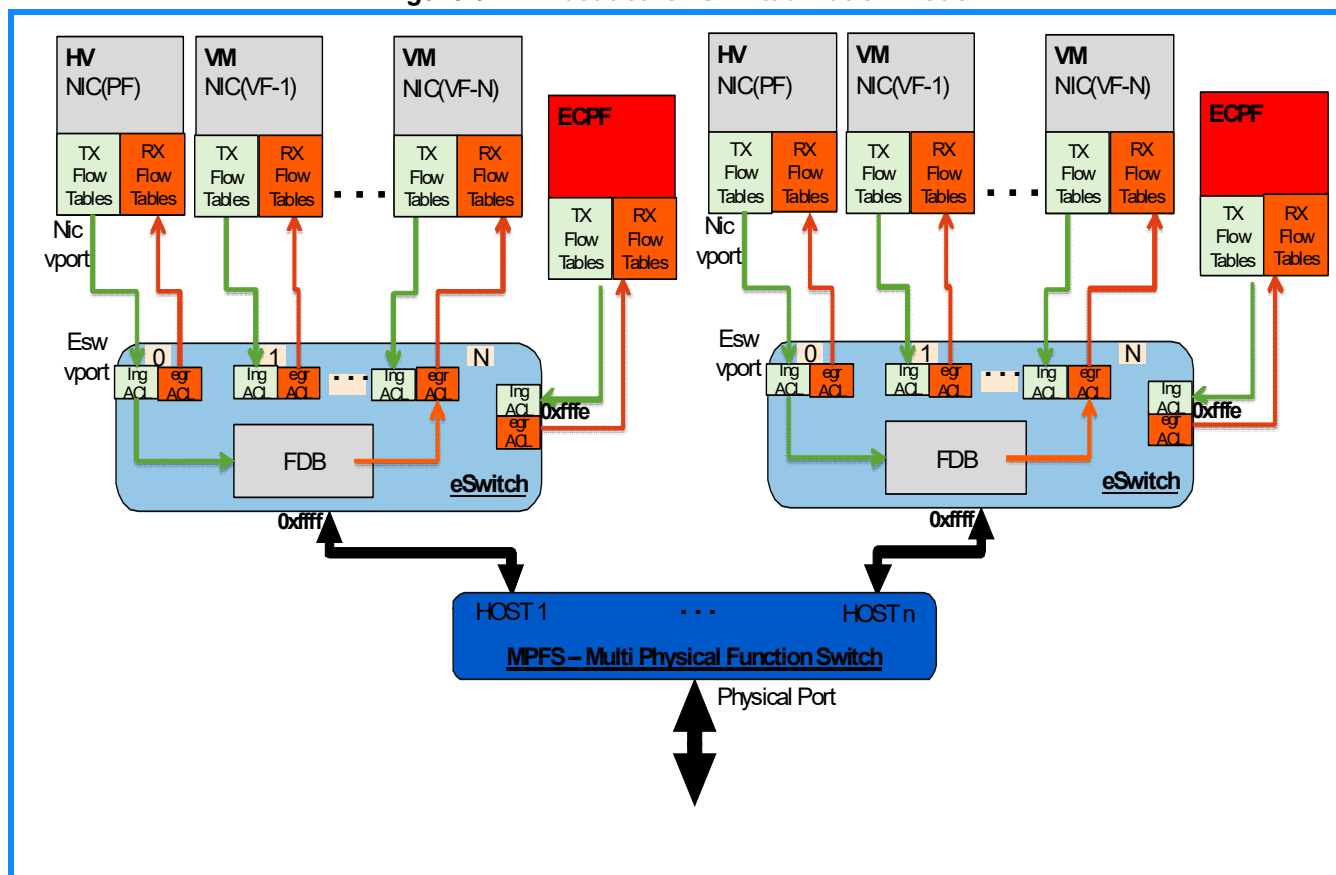


14.5.2 Model 2 - Embedded CPU (EC)

In this model, ARM processors of the BlueField device behave as an Embedded CPU (EC) and part of the external hosts virtualization model.

As shown in Figure 57, “Embedded CPU Virtualization Model,” on page 848, an Embedded CPU physical function (ECPF) is added to each e-switch on vport 0xffff. INIT_SEGMENT.embedded_cpu field indicates if the function is on the Embedded CPU or the external host (See Table 14, “Initialization Segment,” on page 234).

Figure 57: Embedded CPU Virtualization Model



In this model, where ECPF is presented, ECPF takes over most of the responsibilities of the traditional PF running on the external host. It takes over the e-switch management and control responsibilities instead of the PF, including memory pages supplying for the PF and VFs instead of the PF on the traditional external host (See Section 14.1, “Ethernet Virtualization,” on page 838).

The responsibilities that ECPF takes over instead of the PF:

- Owns, controls and manages all e-switch parts:
 - FDB.
 - Ingress and Egress ACLs.
 - State of esw-vports and uplink (not NIC vports that each function still manage its vport).
 - esw_vport_context.
 - Handle NIC vport change event. See Section 14.1.6, “NIC Vport Change Notification,” on page 844
- Takes e-switch traffic by default instead of PF.
- Provides and manages the ICM host memory required for the HCA to store various contexts for it self, the PF and VFs belong the e-switch it manages. See Section 29.3.1, “QUERY_PAGES - Query Device Free Memory Pool Status,” on page 1660.

- Inform the device of the L2 addresses that belong to the vNICs on the e-switch it manages.
- Perform `ENABLE_HCA` for PF to enable it performing initialization flow.

PF will still responsible for:

- Control part of the `nic_vport_context` parameters:
 - Permanent MAC
- PCI and SRIOV configurations
- Perform `ENABLE_HCA` for VFs.

14.5.3 Retrieve Host Information

An embedded CPU function - ECPF (`INIT_SEGMENT.embedded_cpu`) needs to retrieve information about the external host it controls, like host identifier, PCI BDF and number of virtual function. For that, the `QUERY_ESW_FUNCTIONS` command can be used (See Section 29.29, “`QUERY_ESW_FUNCTIONS` - Query e-Switch Functions,” on page 2140).

As part of these parameters may be changed dynamically, SW on ECPF should register for `ESW_FUNCTIONS_CHANGED` event (See Section 8.20.14.4, “`ESW_FUNCTIONS_CHANGED` Event,” on page 503) to get an indication whenever at least one of the parameters is changed. After getting this event, SW need to perform `QUERY_ESW_FUNCTIONS` to get the new information.

14.6 InfiniBand Network Virtualization

InfiniBand Virtualization allows a CA to share its physical link with numerous transport endpoints, referred to as Virtual HCAs (vHCAs). vHCAs have their own transport resources, such as QPs, CQs, SRQs, MRs, and PDs, and expose an independent Verbs consumer interface.

vHCAs connect to the network through an array of light-weight entities called Virtual Ports (VPorts), which share the physical port link. The SMI is owned by the physical port. Each VPort has a unique set of transport attributes: GID tables, P_Key tables, and port capabilities. In addition, each VPort has its own PortState (the logical link state). An HCA that supports virtualization is presented to the InfiniBand subnet as a standard CA with virtualization support. The device does not comprise an embedded IB switch even though packets may be relayed between vports.

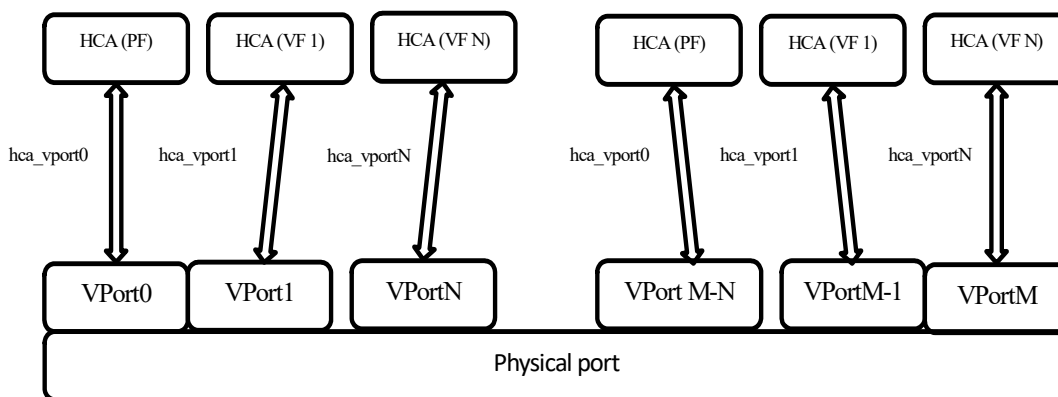
To maintain backward compatibility, the first vport on each physical port mirrors the physical port GID and P_Key tables, and device capabilities. A vHCA that is associated with `vport0` will be full compatibly with legacy SMs, legacy SW running over the VHCA, and legacy SW running on other nodes in the fabric.

Each PCI function (PF or VF) represents a vHCA. Each vHCA virtual port is mapped to an InfiniBand vport, as shown in Figure 58, “IB Virtualization Model,” on page 850. The mapping is arbitrary and determined by the device, as the InfiniBand management is agnostic to it (the InfiniBand specification has no notion of hosts or PCI functions).

VPorts may either be LID- or GID-based. LID-based VPorts are assigned a unique LID by the SM, and thus may send and receive packets without a GRH. GID-based VPorts rely on another

VPort for their LID, and may only send and receive traffic with a GRH. In Multi-Host and SRIOV environments, PF VPort types are determined by non-volatile configuration. VF VPort types are determined by their PF. VPort0 is LID-based and is always associated with the physical port LID.

Figure 58: IB Virtualization Model



14.6.1 Virtualization Support

IB Virtualization is used by multi-host and SRIOV configurations, and supported when `HCA_CAP.ib_virt==1`. SW must exclusively use `vport` commands (See [Section 29.31, “Vport Commands,” on page 2154](#)) to query or modify port state. Virtualization does not depend on suitable SM support. However, a virtualization-aware SM is required to enable `vport` traffic and control their PortState (see PortState section below).

14.6.1.1 Driver Operation

The standard driver initialization and teardown flows also apply when IB Virtualization is present.

However, during operation, GID-base virtual ports must always use a GRH when sending traffic. This distinction is provided by executing `QUERY_HCA_VPORT_CONTEXT` and inspecting the `grh_required` bit. SMPs may be sent and received only if the `has_smi` bit is set, which indicates that the vHCA port is associated with VPort0. Raw Ethernet and Raw IPv6 support is indicated by the `has_raw` bit. (See [Section 29.31.7, “QUERY_HCA_VPORT_CONTEXT – Query HCA Vport Context,” on page 2166](#)).

14.6.1.2 SRIOV Controls

When SR-IOV support is enabled, PF drivers can control the identity, Vport type (LID or GID-base) and PortState policy of their VFs using the `MODIFY_HCA_VPORT_CONTEXT` command (Section 29.31.8, “`MODIFY_HCA_VPORT_CONTEXT` – Modify HCA Vport Context,” on page 2167). A PF may configure LID-based VPorts for its VFs (`grh_required = 0`) only if its own VPort is LID-based (`grh_required == 1`).

VF identity changes take effect when the VF driver is loaded. In addition, PFs may query all the VPort properties of their VFs using the corresponding vport commands. Per-vport statistics may be obtained by `QUERY_VPORT_COUNTER`.

To access VF VPorts, the PF driver must first enable SRIOV. VFs are specified by their logical index, which is passed to the `vport_number` field in the VPort commands. Only visible VFs may be configured; an attempt to configure a VF that exceeds the `NumVFs` field in the SRIOV capability will fail with a `BAD_RESOURCE` status.

To use SRIOV, the PF driver must first complete the standard initialization flow. Specifically, it must execute `CREATE_EQ` with the Page Request event. This EQ reports VF page requests that the PF driver is required to service. Subsequently, VFs may be configured according to the following flow:

- Enable the SRIOV capability with the desired number of visible VFs.
- Execute `ENABLE_HCA` for the visible VFs. This allows VF drivers to issue commands to their VFs.
- Execute `MODIFY_HCA_VPORT_CONTEXT` to configure a VF's identity and port-state policy.
- (Re)load the VF driver.

To disable SRIOV, the PF should conduct the following flow:

- Execute `DISABLE_HCA` for all visible VFs.
- Wait until all VF pages are returned by Page Request events, and reclaim them by executing `MANAGE_PAGES`.
- Disable the SRIOV capability.

Afterwards, the PF driver may unmap the Page Request event by executing `DESTROY_EQ` for the associated EQ, and complete its own shutdown flow. The PF driver is allowed to shut down while keeping the SRIOV capability enabled. Nevertheless, it must still execute `DISABLE_HCA` to reclaim all VF pages beforehand.

14.6.2 vHCA Properties and Resources

Each function is associated with its own vHCA. `QUERY_HCA_CAP` and `QUERY_ADAPTER` commands refer to the device limitations, capabilities and properties of the specific function and vHCA. All commands that manipulate transport resources (e.g., `CREATE_QP`) refer to resources of the corresponding vHCA.

14.6.3 Virtual Ports

Each function is connected to a single vport. Consequently, a function does not provide a vport number when accessing its own vport. A PF may configure the virtual ports of its VFs by providing the logical VF numbering (a 1-based index from the first enabled VF to the last) in vport commands.

Note that vport numbers used to configure vports may differ from the vport numbers visible to IB MADs. The mapping between functions to IB vport numbering is arbitrary, and should be treated as an implementation specific detail.

14.6.3.1 Vport Properties

Vports have their own GID table, P_Key table, PortState, and capabilities. These objects may be queried and modified using the following commands:

- QUERY_HCA_VPORT_CONTEXT
- MODIFY_HCA_VPORT_CONTEXT
- QUERY_HCA_VPORT_GID
- QUERY_HCA_VPORT_PKEY

Note that the P_Key table is completely controlled by the SM. PF drivers have control over their VF GIDs, Node GUID, and PortState, as described below.

14.6.3.2 Port State

Only the physical port has a physical port state (PortInfo.PortPhysicalState). However, both the physical port and all vports have a logical port state (PortInfo.PortState). The vport logical state is independent of the logical state of the physical port. Vports allow packets to traverse them according to their state, similar to physical ports. For example, a vport in the 'Down' state cannot receive or send any packets. Note that vports in the 'Active' state may exchange packets between them even if the PortState of the physical port is 'Down'.

The PortState of virtual ports depends on whether Virtualization was enabled by the SM. Upon power up of the physical HCA, Virtualization is disabled. The SM may enable Virtualization by setting the *VirtualizationEnable* bit in the 'VirtualizationInfo' attribute. This setting persists until the SM explicitly resets the *VirtualizationEnable* bit or the physical HCA is rebooted. The current Virtualization state is reported by the *sm_virt_aware* bit returned by the QUERY_HCA_VPORT_CONTEXT command.

14.6.3.2.1 Virtualization Disabled

When virtualization is disabled, the PortState of vport0 mirrors that of the physical port. Any state transitions of the physical port PortState is immediately replicated vport0. The PortState of other vports is 'Down'.

14.6.3.2.2 Virtualization Enabled

Upon enabling Virtualization, the PortState of vport0 is equal to the PortState of the physical port, while the PortState of other vports is 'Down'. Subsequent PortState transitions, for both the physical port and vports, are governed by both the device and the SM. The device may trigger a transition from any state to 'Down', and from 'Down' to 'Initialize'. The SM controls state transitions as long as the state is not 'Down'.

Physical port PortState transitions to/from the 'Down' state are determined directly by the physical state. However, vport PortState transitions to/from 'Down' are controlled by policy. The PF

may use the `MODIFY_HCA_VPORT_CONTEXT` command to set the policy of the vport to one of the following values:

- Down – the vport PortState always remains ‘Down’.
- Up – if the current vport PortState is ‘Down’, it is modified to ‘Initialize’. In all other states, it is unmodified. The result is that the SM may bring the vport up.
- Follow – follows the PortState of the physical port. If the PortState of the physical port is ‘Active’, then the vport implements the ‘Up’ policy. Otherwise, the vport PortState is ‘Down’.

The above policies apply to vports for which SW has initialized their corresponding device functions (invoked `INIT_HCA`). The virtual port state of functions that are not initialized shall be ‘Down’ at all times.

The default policy of vport0 is ‘Follow’. For other vports, the default policy is ‘Down’.

The following table summarizes how the PortState of a vport is determined:

Table 678 - Vport State Summary

Virtualization Enabled	Policy	Physical Port Port State	Vport0 Port State	VportN Port State (N>0)
NO	*	*	Mirror port state of physical port state	DOWN
YES	DOWN	*	DOWN	DOWN
YES	UP	*	SM Configured	SM Configured (*)
YES	FOLLOW	NOT ACTIVE	DOWN	DOWN
YES	FOLLOW	ACTIVE	SM Configured	SM Configured (*)

Note: (*) GID-based VPorts depend on LID-based VPorts for their LID. In case a GID-based VPort is either Armed or Active while the associated LID-based VPort is not (i.e., it is either Down or Init), the device transitions the GID-based VPort to Init state. Conceptually, this transition goes through the Down state, therefore a Trap indicating the PortState change must be issued.

The `QUERY_HCA_VPORT_CONTEXT` command provides a snapshot of the current values of the port state in the ‘port_state’, ‘vport_state’, and ‘port_physical_state’ fields.

14.6.3.3 Identity

An HCA identity is comprised of its SystemGUID, NodeGUID, and GUID0 (the first index of the GUID/GID table). PFs obtain their identity from the FW image or NV configuration tools.

VF identities are determined by their PFs, using the `MODIFY_HCA_VPORT_CONTEXT` command.

VF identities have two states:

- Administrative state – the values provided by the last `MODIFY_HCA_VPORT_CONTEXT` invocation.
- Operational state – the last value of the administrative state when the VF was not initialized by its driver. From the moment that the driver invokes the `ENABLE_HCA` command until it has called the `DISABLE_HCA` command, the operational state does not change.

The operational state determines packet relay, the identity presented by `QUERY_HCA_VPORT_CONTEXT`, and the values exposed to MADs. This ensures that a VF does not change its identity while it is operational. Consequently, to change a VF identity, the hypervisor must first program the new administrative state, and then re-initialize the VF driver for the changes to take effect. Note that re-initialization also transitions the VF's vport to the 'Down' state, so the Infiniband management always sees a consistent vport state.

14.6.3.4 Statistics

Conceptually, virtual ports maintain unique packet/byte statistics but share the error statistics of the physical port. Thus, virtual ports statistics should be constructed by invoking `QUERY_VPORT_COUNTER` to obtain per vport packet/byte counters, and inspecting the `PPCNT` register to obtain error counters. The physical port packet/byte counts should be ignored.

When IB Virtualization is not used, the vport byte/packet count shall be identical to those of the physical port. Thus, SW should always present statistics for vports. However, if explicit statistics for the physical port are required, such uplink statistics reported by PFs in SRIOV, only the counters returned by `PPCNT` should be used.

In certain cases, functions may not be allowed to observe the physical port counters for security and isolation resources. For example, VFs in SRIOV configurations, or any function in multi-host configurations. In this case, reading the `PPCNT` shall return all counters with a value of 0; SW remains unchanged.

The following table summarizes the commands to obtain statistics in various cases:

Table 679 - Statistics Summary

	Single Host			Multi-Host		
	SRIOV Disabled	PF	VF	SRIOV Disabled	PF	VF
VPORT	Vport counters + PPCNT	Vport counters + PPCNT	Vport counters + PPCNT	Vport counters + PPCNT	Vport counters + PPCNT	Vport counters + PPCNT
PHY PORT	NA	PPCNT	NA	NA	PPCNT	NA

Both physical port and virtual port counters are visible to InfiniBand management via MADs, per the IB specification and the Virtualization Annex.

14.6.4 Special QPs

Each vHCA may create all special QPs. However, only vport0 may send and receive traffic using SMI, Raw Ethertype, and Raw IPv6 QPs. For other vports, sent traffic is silently dropped, and no traffic is ever received. The GSI is operational for all vports.

A function can observe if it can conduct traffic via special QPs by executing QUERY_HCA_VPORT_CONTEXT. The *has_smi* and *has_raw* bits indicate access to the SMA and Raw Ethertype/IPv6 QPs, respectively.

14.6.5 Packet Relay

14.6.5.1 Unicast

Packet relay is defined by means of steering and loopback as follows. All vports shall receive any traffic that contains a GRH and matches the vport GID table. vport0 shall receive all other traffic (including SMI, Raw IPv6, and Raw Ethertype).

Every packet sent by a vport shall be looped back to that vport if the DLID refers the physical port LID and the DGID matches the vport GID table. Every packet sent by the physical port shall be looped back if the DLID matches the physical port LID. In addition, every packet sent with the force loopback indication shall be looped back to the sending vport.

14.6.5.2 Multicast

Any multicast packet received or sent by the physical port is forwarded to all QPs attached to the corresponding MGID for which the corresponding vport is in the ‘Active’ PortState. Multicast packets transmitted by QPs whose vport is not in the Active PortState are relayed outside the vport.

14.6.6 Asynchronous Events

14.6.6.1 Affiliated Events

Affiliated Asynchronous Events and Errors are delivered solely to the vHCA that owns the affiliated resource.

14.6.6.2 Unaffiliated Events and Errors

A vHCA receives a Port_State_Change event in the following cases:

- The physical or vport ‘PortState’ transitions to the LinkActive state
- The physical port or vport ‘PortState’ becomes unavailable, i.e., enters one of the following states:
 - Down, Initialize and Armed.

- A change in the following physical port attributes:
 - MasterSM Lid, MasterSM SL, SubnetTimeout, LID change
- A change in the following vport attributes:
 - GID Table, P_Key table
- Client Reregister was invoked on either the physical port or vport

A vHCA receives an Internal_Error event whenever the vHCA state or the whole device state is undetermined.

14.7 vNIC Monitoring

SR-IOV technology allows assigning device resources directly to VFs. In cases where the VFs are not trusted, the Hypervisor will require means to monitor the VFs behavior and utilization of the device resources.

The diagnostics collected by the device regarding the VF behavior can be collected using the “[QUERY_VNIC_ENV – Query vNIC Environment](#)” command. It is supported if *vnics_env* field is set in “[QUERY_HCA_CAP – Query Device Capabilities](#)”.

Statistics for packets that defaulted the NIC Receive Flow Table and discarded is supported if *nic_receive_steering_discard* is set in the same command. Statistics for packets that were discarded because the vport was not in a state to handle them is supported according to *receive_discard_vport_down* and *transmit_discard_vport_down* fields.

14.8 Multi vHCA Mkey

When *HCA_CAP.m_vhca_mk* =1 (on all vHCAs), the device supports sharing Memory Keys across VHCA's set with the same *INIT_HCA.sw_owner_id*.

To enable Mkey sharing, SW shall set *INIT_HCA.m_vhca_mk_child*=1 to indicate that this VHCA will refer to the VHCA specified by *INIT_HCA.m_vhca_mk_parent_vhca_id* for any Mkey access.

Mkey references may be initiated either by local operations or by incoming packets and may also modify the state of the Mkey.

If the driver has set *INIT_HCA.m_vhca_mk_child*=1, it shall not issue any *CREATE_MKEY* command on that VHCA nor any of the following objects associated with Mkey:

- PSV
- PD
- KEK/DEK

Possible use cases of this feature:

- Implement a single multi-port IB HCA over a function-per port device. For example: A dual port IB HCA over a device with two network ports, each represented by its own PCI function. Thus share Mkeys across the two ports.

- Implement a single single-port IB HCA over a multi PCI port device (see the “Multi PCI port” section for details).

14.9 Multi PCI Port

Supporting a network device port connected to the host through multiple PCIe buses may be desired in these two cases:

- The network port speed is higher than that of any available PCIe socket in the system
- NUMA (Non Uniform Memory Access) systems, for improved NUMA system utilization

If `HCA_CAP.m_pci_port=1` (on all vHCAs), the device is capable of driving a single network port via multiple PCIe buses in the following manner:

Manage the port shared resources through one of the PCI buses while managing the resources affiliated with a PCI I/F through the PCI bus they are affiliated by, thus balancing the load over the multiple PCI buses.

The vHCA through which the port shared resources are managed will be referred to as the “port parent vHCA” while other vHCAs associated with the same port that share the port resources with the port parent vHCA will be referred to as “port child vHCAs”.

The partitioning of resources to port resources and PCI resources is as follows:

Port resources:

- Physical port management / debug
 - All Register Access commands
- Network processing
 - GID table / RoCE address table
 - MCG
 - L2 Table (MPFS)
 - Flow Table
 - SNIFFER
 - E-switch
 - Vport
- Congestion Control commands
- LAG
- Special QPs and Management commands
 - QUERY/SET_MAD_DEMUX, MAD_IFC
- Transport I/F
 - RQT

- TD (Transport Domain)
- TIS
- TIR
- Memory keys and their accompanies (if supported by the device)
 - Mkey
 - PSV
 - PD
 - DEK/KEK (Crypto)

PCI resources:

- QP
- DCT
- DCI
- SQ
- RQ
- XRQ
- RMP
- SRQ
- XRC SRQ
- XRC Domain // TODO
- Q_COUNTER
- UAR
- CQ
- EQ
- MSI-X

Notice that the Mkey and its associated resources may be shared across vHCAs too although they are in the category of affiliated with a PCI I/F. See Section 14.8, “Multi vHCA Mkey,” on page 856.

14.9.1 Port parent/child vHCA selection

The driver shall realize, at PCI device probe time, the vHCA association with a port by the port GUID field, as reported in the QUERY_NIC_VPORT_CONTEXT and the QUERY_HCA_VPORT_CONTEXT for an Eth and IB port respectively.

If the driver realizes that a vHCA is associated with the same port as an already probed vHCA, and HCA_CAP.m_pci_port=1 for this vHCA and the already probed vHCA, it may configure it to be a port child vHCA at INIT_HCA by setting:

- `m_pci_port_child = 1`
- `m_pci_port_parent_vhca_id = HCA_CAP.vhca_id` of the parent.

The `INIT_HCA.sw_owner_id` of the port parent vHCA and the port child vHCA must be equal.

Once a vHCA is set as port child vHCA, no port resources (as listed above) commands should be issued on it.

14.9.2 Cross vHCA Referencing

As a consequence of the resource partitioning across port parent and child vHCAs, resources of one vCHA may now refer to resources of another vHCA:

- Port child vHCA resources referring to port parent vHCA resources
 - QP/WQE AV --> GID / RoCE address Table index
 - SQ → TIS
- Port parent vHCA resources referring to port child vHCA resources (TBD ahmad – don't you mean pci resources)
 - TIR --> RQ
 - RQT-->RQ
 - MCG (Multicast Group) --> QP

In case a PCI resource refers to a port resource, the vHCA of that port resource is implicitly the port parent vHCA.

In case a port resource refers to a PCI resource, the vHCA of that PCI resource must be specified by SW.

14.9.2.1 TIR → RQ

For a TIR referring to an RQ by the `TIR.inline_rqn` field, SW shall set:

- `TIR.inline_rqn_vhca_id_valid = 1`
- `TIR.inline_rqn_vhca_id = HCA_CAP.vhca_id` of the vHCA through which the RQ was created

If SW does not set the `TIR.inline_rqn_vhca_id_valid` bit, then the device will assume that the RQ was created through the same vHCA of the TIR.

14.9.2.2 RQT → RQ

For an RQT with one or more RQs residing on another vHCA, SW shall set

- `RQT.vhca_id_valid=1`
- For each RQ in the RQT
 - Set the corresponding entry in the `RQT.vhca_id[]` array to the `HCA_CAP.vhca_id` of the vHCA through which the RQ was created

If SW does not set the RQT.vhca_id_valid bit, then the device will assume that all RQs of the RQT were created through the same vHCA of the RQT.

14.9.2.3 MCG è QP

For an MCG referring to a QP by the qpn field of the ATTACH_TO_MCG command, SW shall set:

- qpn_vhca_id_valid = 1
- qpn_vhca_id = HCA_CAP.vhca_id of the vHCA through which the QP was created.

If SW does not set the qpn_vhca_id_valid bit, then the device will assume that the QP was created through the same vHCA through which the ATTACH_TO_MCG command is issued.

14.9.3 RX IB Unicast vHCA selection

For Raw Ethernet and IB multicast the destination RQN / QPN of an incoming packet is configured by local SW by port resources such as flow tables, TIR, RQT, IB Multicast Groups.

For incoming IB unicast packets however, the destination QPN is specified in the packet BTH header. It is the device responsibility, not SW, to route incoming IB unicast packets, based on the packet BTH Destination QPN, to the vHCA through which the QP was created (and thus to the PCI bus the QP is associated with).

15 LAG - Link Aggregation

The device supports link aggregation of its physical ports, in which the two physical port can be aggregated to one port. SW may be allowed to create and manage LAG. This feature is supported only when `HCA_CAP.num_lag_ports>1`, See Table 1571, “HCA Capabilities Layout,” on page 1671.

If `HCA_CAP.lag_master=1`, SW is responsible to enable and manage the LAG using the `CREATE/MODIFY/DESTROY LAG` commands See Section 29.4, “LAG Commands,” on page 1849. In case the SW decides to aggregate multiple ports, each represented by a different PCI function, the driver must select the lowest PCI function aggregated as its “LAG master function”. The LAG management commands are allowed only via this “LAG master function”.

Following LAG creation:

- The driver QPs/TIRs will accept incoming traffic from all LAG ports.

Outgoing traffic will go through the port specified by the QP/TIS `lag_tx_port` affinity field unless that output port was remapped to another port by the `CREATE/MODIFY_LAG` commands. This LAG TX port remapping does not apply to TISs with the `strict_lag_tx_port_affinity` field set.

The LAG master may use the `CREATE_VPORT_LAG` command to merge the flow steering lookup of all LAG vports into the master’s flow steering lookup. This includes implicit RDMA unicast flow steering by packet BTH QPN.

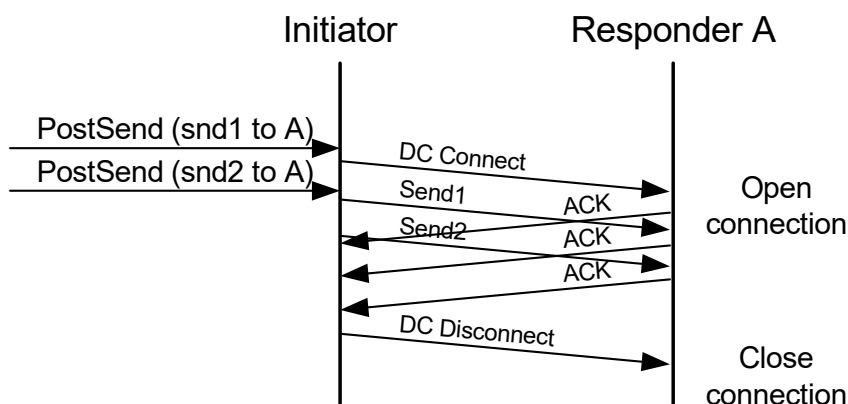
The LAG master may use the special vport LAG demux flow table (see Table 2099, “`CREATE_FLOW_TABLE` - Input Structure Field Description,” on page 1987) to allow packets coming from vports not natively associated with the LAG master to “jump back” to the `NIC_RX` root flow table of their source vport

Following `CREATE_VPORT_LAG` a packet coming from any LAG vport will be counted in the LAG master vport counters. A packet that “jumps back” to its native vport (by the vport LAG demux flow table) will be counted also on its native vport counters (i.e it will be counted on both vports).

16 Dynamically Connected Transport Service

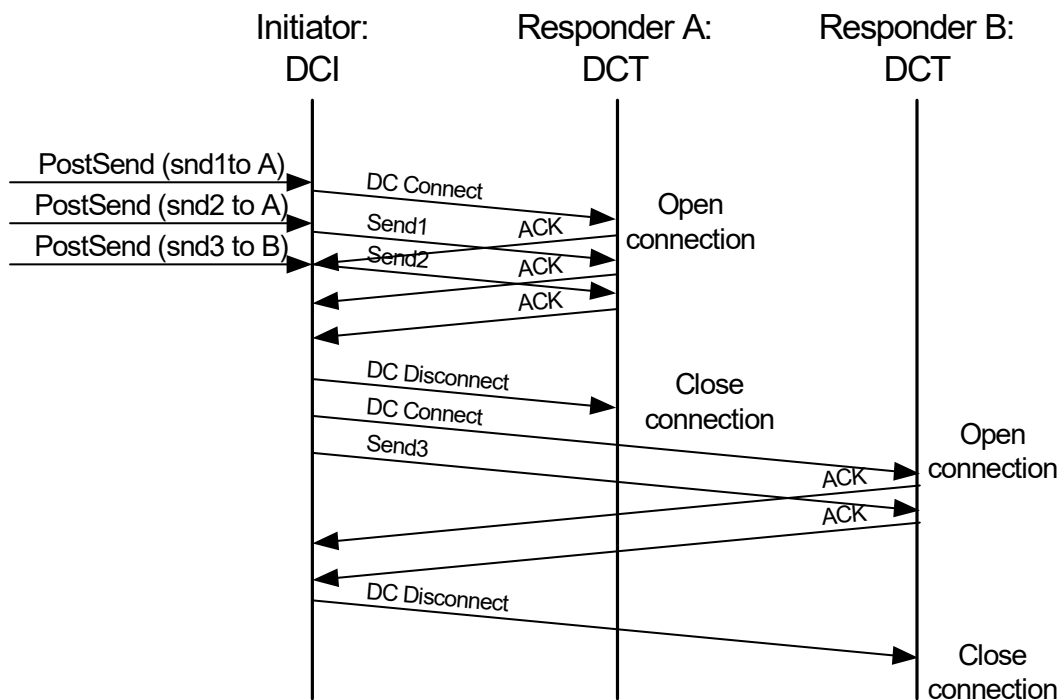
The Dynamically Connected (DC) Transport Service provides a datagram-like model that allows a DC QP to target multiple remote processes in multiple remote nodes. As far as reachability is concerned, the DC model is somewhat similar to the Unreliable Datagram (UD) model in the sense that each WR submitted to the DC SQ carries the information that identifies the remote destination. DC contexts are then dynamically tied to each other across the network to create a temporary RC-equivalent connection that is used to reliably deliver one or more messages. This dynamic connection is created inband and pipelined with the subsequent data communication thus eliminating the cost associated with the 3-way handshake off the Connection Manager protocol used for connecting RC QPs. When all WRs posted to that remote network address are acknowledged, the initiator sends a disconnect request to the responder, thereby releasing the responder resources (Figure 59).

Figure 59: Basic DC Connect-data-disConnect Sequence



WQEs posted to one SQ may target multiple different destinations. When the initiator reaches a point in its SQ where the next WQE is destined to a different target (for example another process - possibly in a different node), the previous dynamic connection is torn down and the new one is established as illustrated in Figure 60.

Figure 60: Dynamically Connected Data Transfer



The DC transport service can be described at a high level as an extension to RC that adds inline connect and disconnect primitives while preserving the reliability semantics and capabilities of the RC protocol. While connected, a DC Initiator provides the exact same services as an equivalent RC one. All of the RC operations are supported and error semantics are identical.

The DC transport service is implemented via new QP type in the send side which called DCI - DC Initiator, and a new object in the receive side called DCT - DC Target. By virtue of its dynamic connection characteristics, a DC Initiator cannot know a priori the numeric identifier of the specific connection that will be handling its requests. Therefore, in contrast with RC packets that target a specific DestQP, DC request packets target a DCT (Dynamically Connected Target) which is the logical entity that is associated with DC connections out of which one is picked for handling the dynamic connection in question at connection time. Once the dynamic connection has been established, all subsequent request packets of that connection (as identified by their source address and source QPN) are handled by the specific connection that was created for that purpose at connect time.

DC connections are created and torn down by means of new connect and disconnect packets that are sent inband just like the regular data packets that are sent while the connection is operational. This explicit connect and disconnect model keeps the DC operation close to the RC model.

The connect packet carries all the information required to establish the temporary connection between the DC Initiator and the target node. Upon reception of a connect request packet, the destination node attempts to open a new DC connection. If the creation of a new connection is successful, then this connection becomes ready to process inbound request packets. If there are no available resources to handle the connection request, a NAK is sent back to the Initiator¹. If a

violation is detected (like access_key violation), the connect message is silently dropped at the responder. See TBD “section for handling key violation”

Disconnect requests are sent by the Initiator when a DC connection is no longer required, thus freeing the connection resources in the DCT and allowing further subsequent connections to it from another Initiator.

Error semantics are those of RC QPs with the addition of a few DC specific ones such as those that describe problems in the ability to create or tear down a connection.

Protection and isolation is implemented via *DCT_Access_Key* that is carried in each connect request. Each DC Target is configured (value and mask) to accept connect requests only if *DCT_Access_Key* field in connect request matches (after mask) the respective field configured on the DC Target. The mask will enable comparing (unmasked) 64 bits, 32 bits, 16 bits or 8 bits of the key. The bits being compared are the msb bits of the key.

If the high order bit of the *DCT_Access_Key* is set, the outgoing packet shall contain the *DCT_Access_Key* from the QP Context. If the high order bit of the *DCT_Access_Key* is not set, the outgoing packet shall contain the *DCT_Access_Key* from the WR.

16.1 DC Transport Service Definition

DCI SQ is used to post DC WRs. DCI SQ provides same execution capabilities as RC SQ and can be used to execute data transfer operation as well as WRs that do not result in packets being sent on the wire (for example UMR, etc.).

Each Data transfer descriptor (WQE) contains the address vector of the remote peer (network address and DCT) in addition to the parameters used to post WRs on RC SQs; it is schematically illustrated in Table 680.

Table 680 - DC Descriptor Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0		seg
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											ctrl
Control segment - same as RC WQE																														A		
Address Vector segment - same as UD segment, DestQP field specifies DCT, DCT_Access_Key is replacing Q_Key																														Radr		
Remote Address segment (if RDMA)																														Data		
Gather list or atomic data																																

Note that “wire-less” WQEs on DC QPs (NOP, UMR, PSV WQEs, Wait, Send En, Receive En, DUMP) will not contain Address Vector. Their format is like for RC QPs. Detailed DC WQE format is described in “Work Request (WQE) Formats” on page 310.

1. Sending NAK if connection request cannot be satisfied is best-effort by the Responder.

16.1.1 DC Packet Format

The DC packet format is that of RC packets with the addition of a DCETH to all¹ packets including the new inband connect and disconnect as shown in [Table 681](#).

Table 681 - DC Packet Format

LR H	GRH [optional]	BTH+DCETH	Extended transport header(s) [optional]	Immedi- ate [optional]	Payload	iC RC	vC RC

Extended transport headers and Immediate are added based on the opcode like for RC packets.

BTH and DCETH for DC packets is shown in [Table 682](#).

Table 682 - DC Transport Header

3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
Opcode								S	E	M	Pad	TVer				Partition Key															
FE	BE							DC Target (DCT) for request, DC Initiator (DCI) for response																				BTH			
A							PSN																								
reserved																												DCETH			
								DC Initiator (DCI) for request, DCT for response																				DCETH			

Opcode field contains transport opcode. Encoding is identical to RC opcodes' encoding with code field (opcode[7:5]) set to 010, hereby usurpating Reliable Datagram opcodes' space. CONNECT/DISCONNECT opcode is 10101 (usurpating the RESYNC opcode), For CONNECT, SE bit is 0 while for DISCONNECT SE bit is 1. Negating connection request (CNAK) is an ordinary NAK packet (that includes DCETH header) with CNAK syndrome (former Invalid RD Request syndrome - 0b01100100).

16.1.2 DC Connection and Disconnection Model

DC contexts are disconnected while Idle. When a data-transferring WQE is posted to the SQ of a DC Initiator context, the HCA sends a connect request to the end-node that is being targeted in the AV of the WQE in question. The connect request identifies the target by means of the DCT that is specified in the posted send WQE and is carried in the DestQP field of the BTH in DC request packets. Immediately after, and back to back with the connect request, the HCA can proceed to send the regular request packets for the posted WQE without having to wait for any confirmation on the success of the connect request. This speculative execution model is a key

1. response packets carry DCETH too so that the Initiator can validate DCT (in sourceQP field of DCETH).

element in the DC model that brings its latency characteristics to par with those of an equivalent RC connection (this is assuming that connect request almost always succeed.

16.1.2.1 DC Connect and Disconnect

The DC Initiator sends a Connect request when a data transfer WQE is posted to its idle SQ or after sending a disconnect request for a previous DC connection upon reaching a WQE that targets a different destination. The Connect packet is sent to the node address and DCT specified in the WQE. Connect Request packet carries information required to establish connection in an additional DCCETH (DC Connect Transport Header), shown in [Table 683](#).

Table 683 - DC Connect Extended Transport Header

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0																	00h-04h
DCT_Access_Key[63:0]																																08h-0Ch						

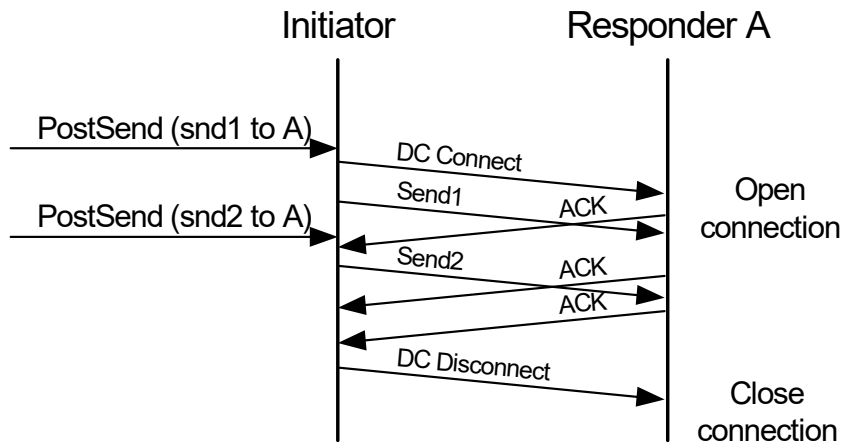
DCCETH will follow the DCETH header.

DCT_Access_Key is used to validate access rights to the DC Target. Connect request is granted if masked key in the connect request matches one configured in the DC Target (connect.DCT_Access_Key & DCT.DCT_Access_Mask = DCT.DCT_Access_Key).

From the DC Initiators' perspective, the sending of a DC Connect request [speculatively] establishes the DC connection and that allows the immediate sending of regular request packets (back to back with the connect request) without having to wait for any kind of response for the Connect itself.

Upon reception of a connect request, the responder node verifies its validity and attempts to create a connection. The created connection becomes tied to the DC Initiator identified by its source QPN and network address, hereby establishing the connection. At that point, this connection becomes ready to process inbound requests including those that the Initiator may have sent back to back with the connect request as illustrated in [Figure 61](#).

Figure 61: Successful DC Connect



The PSN in the DC connect packet is unilaterally determined by the Initiator and is used by the Responder to synchronize its first expected PSN for the connection in question. Consequently, the first data packet following the DC connect request will have a PSN that is equal to the PSN in the DC connect request + 1. Disconnect packet will have PSN larger by 1 than the former packet.

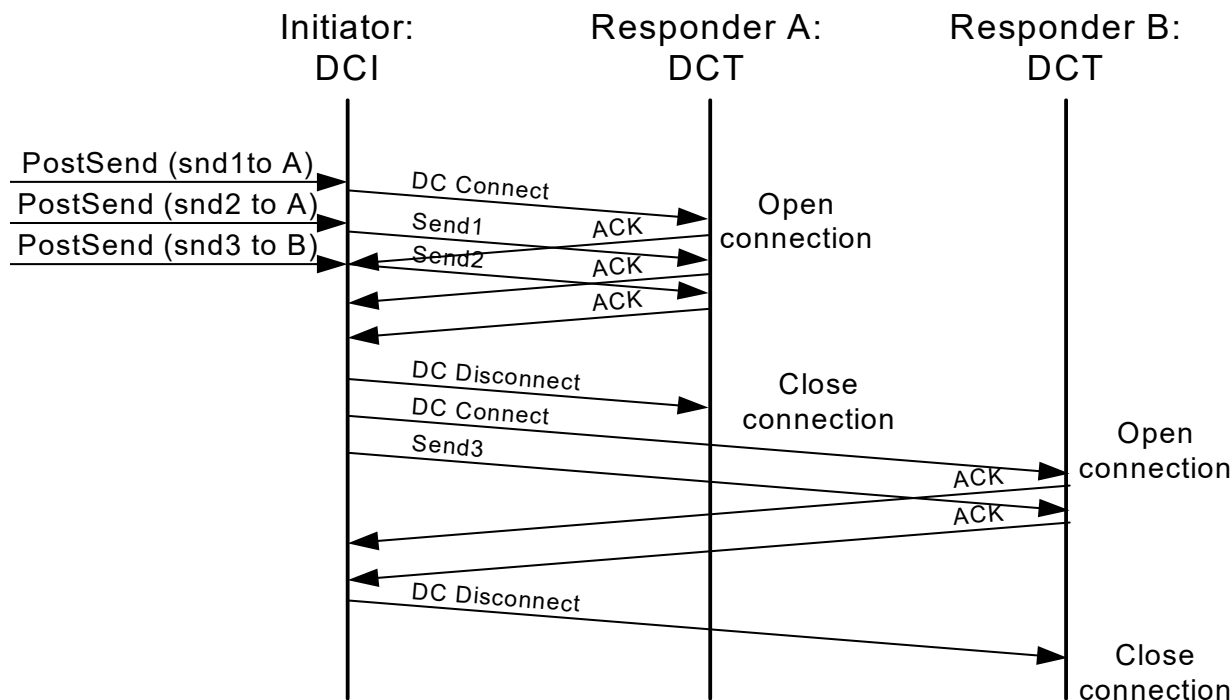
In order to allow speculative (pipelined) execution of data request, end-to-end parameters that are normally being negotiated during RC connection establishment are globally and uniformly configured for all DC Initiators and Responders (for example number of outstanding RDMA-read requests, timeouts etc).

DC Initiators follow RC rules for generating PSNs thus allowing RC reliability and semantics. DC packets received after the connection was established are processed as described in “[DC Data Packets](#)” on page 872.

A DC Initiator will send a disconnect request when it detects that it has no more pending WQEs to execute or when the next WQE targets a different destination than the one it is currently connected to. However, prior to sending the disconnect, the Initiator will wait until all its outstanding messages have been completed¹ (see [Figure 62](#)). In this manner, the Initiator guarantees that the DCT will have its connection context available to process repeated requests (if necessary) thus preserving RC semantics. The DC disconnect packet does not generate a response and the Initiator does not wait for one.

1. It may happen that while waiting for outstanding messages to complete new WQEs are posted that target the currently connected destination in which case the Initiator may not send the disconnect and proceed with the subsequent messages.

Figure 62: DC Disconnect + Connect to Other Destination



DC Disconnect packets are processed by the active connection based on regular PSN rules. It is expected that a valid DC Disconnect will always have the expected PSN since the Initiator would not have sent it if it had not previously received acks for all previous requests¹. So any Disconnect that does not match the expected PSN can be silently dropped.

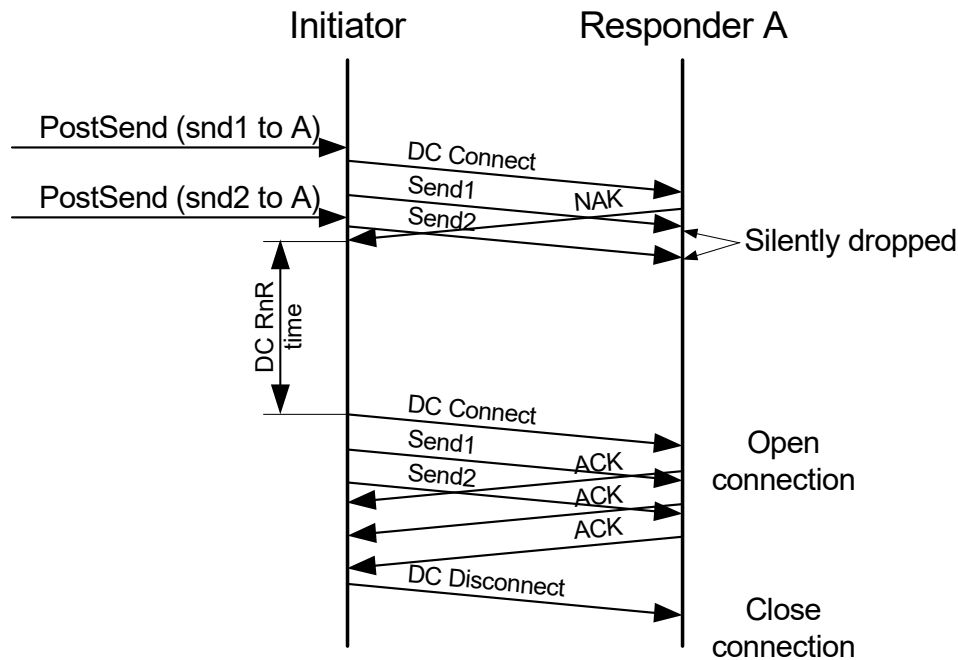
16.1.2.2 DC Connection Denial (Connection NAK)

A DC connection request could be denied if the DCT does not have enough resources to honor it (for example all DC Responders are currently in use). In this case, the target may send a connection NAK (CNAK) to the Initiator^{2,3}. No state is kept on the responder for denied connect requests and subsequent data packets will not match any active DC Responder context and will be silently dropped. After receiving a CNAK, the DC Initiator will wait for a configurable time and then proceed to retry as depicted in [Figure 63](#).

Notwithstanding the above, in order to achieve good performance, the DC transport service should be operated so that the CNAK scenario becomes extremely unlikely.

1. By definition, the Initiator never sends subsequent packets after the Disconnect. So the Disconnect PSN can never be in the “repeated request” range. Also the Disconnect PSN could never be OOS because that would imply that there is a prior request that had not yet arrived to the responder and therefore could not have been completed by the Initiator.
2. CNAK is “best effort” from responder point of view. It is a performance optimization since Initiator will timeout if it does not get a response.
3. CNAK to connect requests could be wasteful in the fabric because the Initiator pipelines subsequent data request that will all get dropped after having consumed transmission resources (at the endnode and in the fabric).

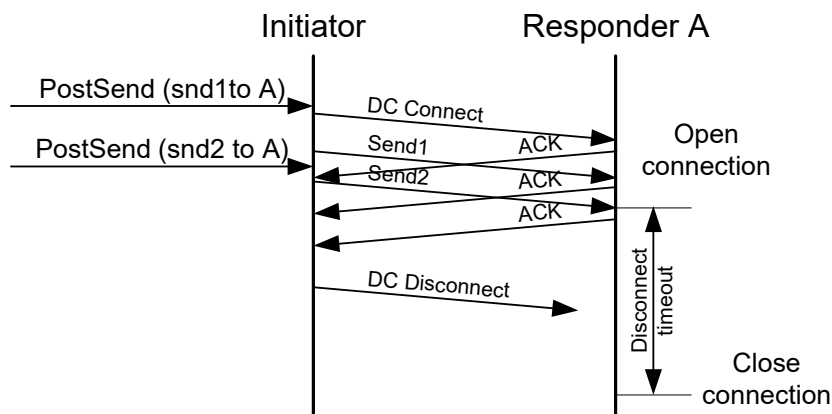
Figure 63: Denied DC Connect Request



16.1.2.3 Lost Disconnect

The DC disconnect request is used to release the connection resources on the target. For the rare case where the disconnect is lost, the target implement a disconnect timeout that eventually close the connection releases its resources after enough time has passed since the last activity was seen on the connection. This case is illustrated in [Figure 64](#).

Figure 64: Disconnect Timeout



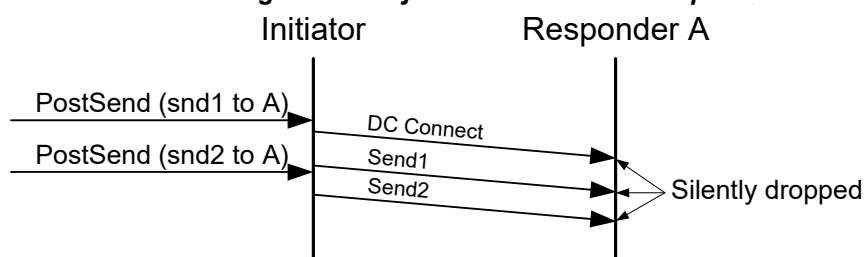
The disconnect timeout should be long enough to ensure that no more repeated requests can potentially arrive on this connection from the Initiator (for example if Initiator timeout is T and

number of retries is N, the timer value must exceed $2 \times T_{TxN}^1$). Otherwise, the Initiator may not be able to complete its outstanding requests and will be forced to complete them in error.

16.1.2.4 Connect Request Reject

In the process of receiving a connect request, the responder will verify its validity making sure that *DCT_Access_Key* in the packet must match that of the DCT. If *DCT_Access_Key* in connect request matches one configured in the DC Target, but no connection resources associated with DCT are available, responder will deny connection request. If *DCT_Access_Key* in the connection request does not match the one configured on the DC Target, the connection request is silently rejected - no response is sent to the Initiator (Figure 65).

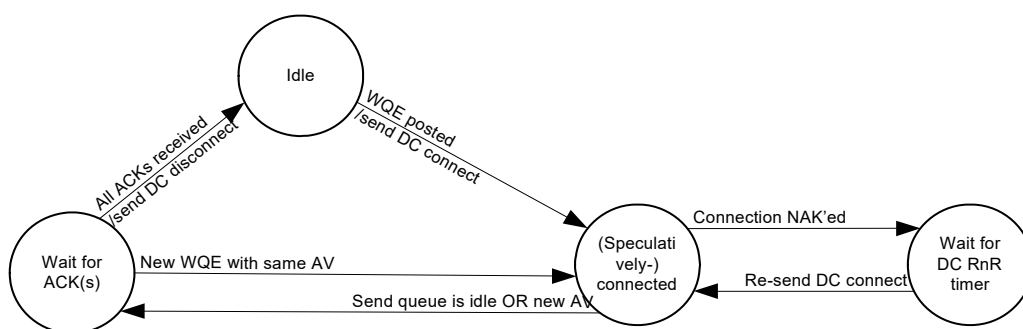
Figure 65: Rejected DC Connect Request



16.1.2.5 DC Initiator Context State Machine

The DC Initiator connection state machine is schematically shown in Figure 66.

Figure 66: DC Initiator Connection State Machine



1. The timeout needs to consider sender scheduler time constants if they are different from those of responder.

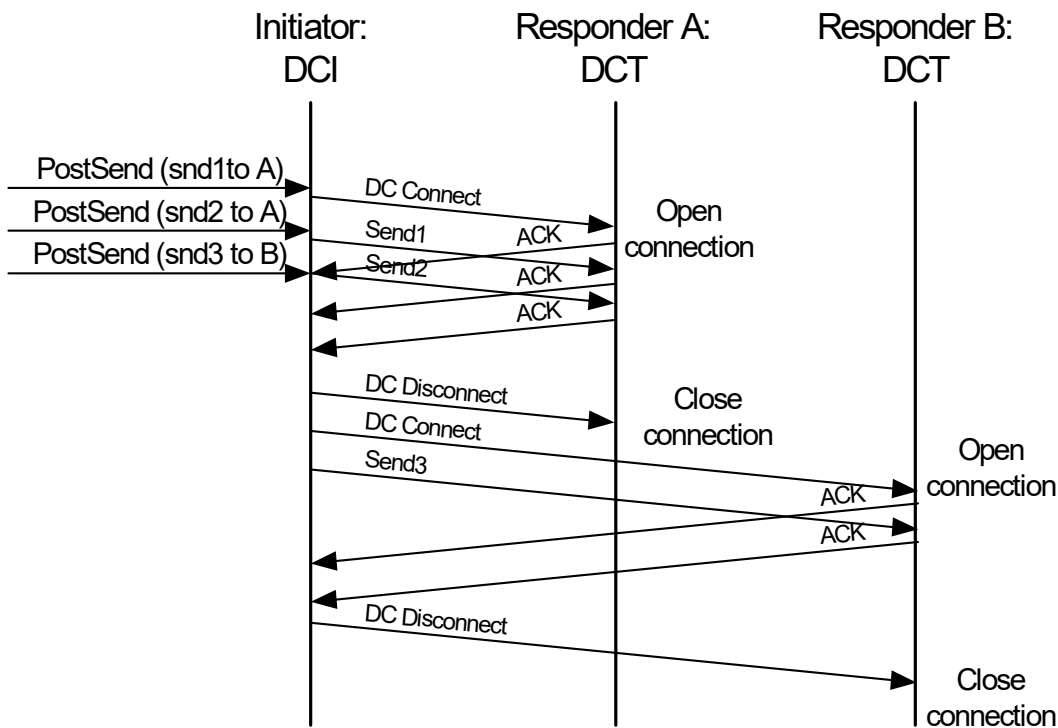
Idle is an initial state of each DC Initiator (SQ). Upon posting data transfer WR, DC Initiator sends **Connect** request to remote peer whose address vector is specified in WQE and moves to (*Speculatively-Connected*).

In *Speculatively-Connected* state, DC Initiator sends data request packets and implements all RC transport and protection checks in action (re-send in case of transport timeout). Error condition that would normally cause RC QP closure (access violations, re-tries exceeded etc.) and puts the QP to ERROR state. If Connection NAK is received, DC Initiator waits for Connection RnR NAK timeout and re-tries the connect request.

Upon reaching end of SQ or upon reaching address vector that does not match the one of current connection, Initiator moves to *Wait for ACKs* state. In this state, DC Initiator implements all transport and protection checks and actions and after all outstanding messages are successfully ACK'ed, it moves to *Idle* state. Error behavior is same as in *Connected* state.

The disconnect request is never acknowledged by the responder. This is intentional and meant to optimize for the typical case where disconnect packets are not lost and reach the destination in due time. This allows for a subsequent connect request (upon detection of a different destination on the next WQE) to proceed right back to back with the disconnect request from the previously connected responder - without having to wait for a response (Figure 67).

Figure 67: Disconnect Followed by Connect to Another Destination



In the rare case that the disconnect does not reach the responder, a responder timeout mechanism will eventually release the DC Responder context as described in “[DC Connect and Disconnect](#)” on page 866.

16.1.3 DC Data Packets

Data packets can be sent immediately after the connect request and are guaranteed to arrive in order (after the corresponding connect) by means of them being sent using the same link layer parameters as the connect itself.

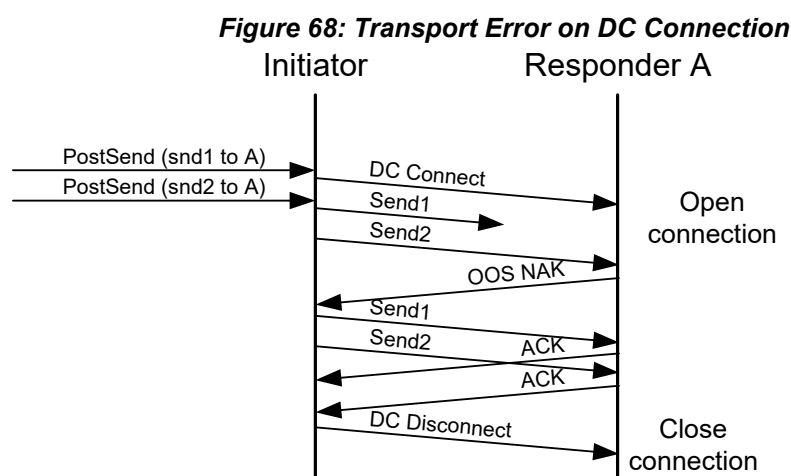
The SourceQP number in the DCETH and the source network address are used by the responder to find the specific connection resource in the target that is currently serving this connection. If no such connection resource is found then the packet is silently dropped.

The DestQP field in the BTH of the response packets carries the value received in the SourceQP field in the BTH of all inbound DC requests, explicitly identifying the Initiator QP number.

The SourceQP field in the DCETH of the response packets carry the DCT that is handling the connection and can be used to silently drop spoofing attempts at the Initiator.

16.1.4 Data Packets Error Behavior

Transport and access violation errors on DC connection are handled same way as they are handled on RC connection. One example is shown in [Figure 68](#).



16.2 DC Control Objects

DC Transport Service composed of two objects: DC Initiator (DCI) and DCT (DC Target).

16.2.1 DC Target

DCT is an object identifying a target of DC operation that can handle DC receive operations.

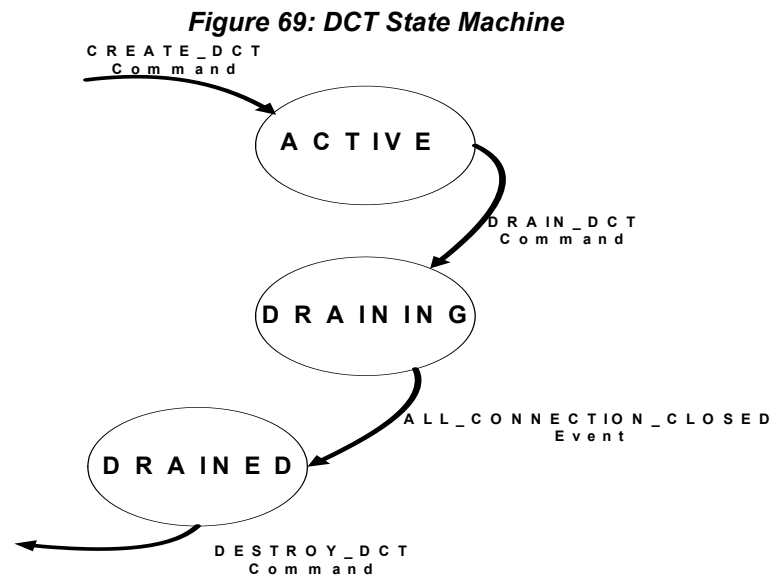
DCT created using CREATE_DCT Command (See [Section 29.30.1, “CREATE_DCT – Create DCT,” on page 2144](#)).

Destroying DCT should be done as described in [Appendix 69, “DCT State Machine” \(page 873\)](#).

a DRAIN_DCT command must be called before destroying a DCT, this will move DCT to a DRAIN state, which close all opened connections and prevent opening new connections on this DCT.

After HW finish closing all opened connections, an asynchronous event called “DCT All Connections Closed” will be generated. See [Section 8.20.8.1, “DCT Events,” on page 480](#)

Just after getting this event software is allowed to call DESTROY_DCT command.

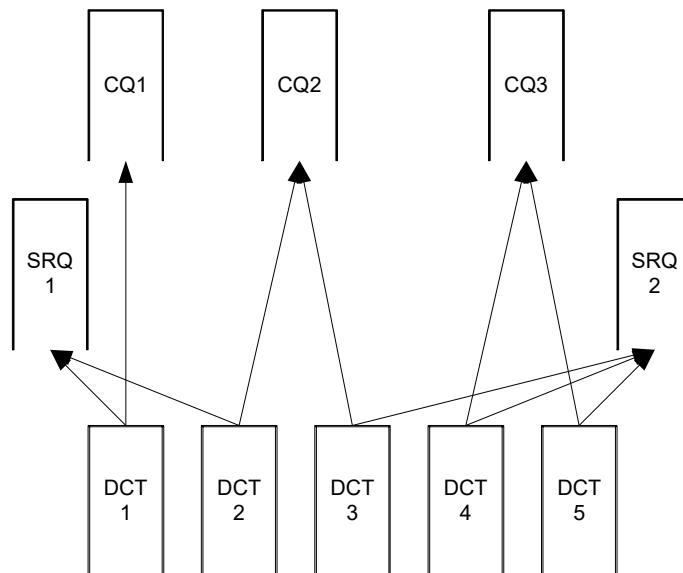


At the creation time, DCT is associated with SRQ to be used to process inbound send requests and with CQ where completion operations on that DCT will be posted¹. [Figure 70](#) illustrates associations of DCTs with SRQs and CQs.

HW will scatter the GRH to the CQE instead of WQE buffer (which is UD default behavior), so if user needs to connect DCT to 128B CQE.

1. Association of DC Responder with CQ enables to share a single pool of receive buffers (SRQs), while enabling the Initiator to control the receive process of the message (through DCT).

Figure 70: Association Between DCTs, Receive Transport and Receive Queue



16.2.2 DC Initiator

DCI is a send part of DC QP.

16.2.3 DC Completion Queue Entries

16.2.3.1 DC Initiator CQE

DC Initiator CQE is format is similar to RC CQE format. For details see [“64B Completion Queue Entry Format Layout” on page 435](#).

16.2.3.2 DC Target CQE

DC Responder CQE is similar to UD receive CQE in a sense that the source address of message Initiator is reported in the CQE. In addition, it includes all information that is normally reported in RC receive CQE. For details see [“64B Completion Queue Entry Format Layout” on page 435](#).

16.2.4 DC CNAK Trace

Device can be configured to dump the Lid address of the received CNAK packets to the memory of the host (DC_CNAK_TARCE_TABLE). This feature is supported only when HCA_-CAP.dc_cnak_trace==1 (See Table 1571, “HCA Capabilities Layout,” on page 1671).

The feature is enabled as follows:

- SW should allocate a single 4KB chunk for single port device or 8KB chunk for dual port device, where the device dump the DC_CNAK_TARCE_TABLE.
- SW must initialize the allocated chunks to 0xff's.

- SW inform the device about this page using SET_DC_CNAK_TRACE command. (See Section 29.30.7, “SET_DC_CNAK_TRACE – Set DC CNAK Trace,” on page 2151).
- Device generate new DC_CNAK_TRACE_ENTRY each time it reach a full entry of CNAKs (received 30 CNAK packets).
- DC_CNAK_TRACE_ENTRY include an incremental sequence_number so SW can follow it to know the validity of each entry.

DC CNAK TRACE TABLE is a 4KB cyclic memory per port as shown in Table 684

Table 684 - DC_CNAK_TRACE_TABLE Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dc_cnak_trace_entry[0]																																00h-3Ch
dc_cnak_trace_entry[1]																																40h-7Ch
dc_cnak_trace_entry[2]																																80h-BCh
...																																...
dc_cnak_trace_entry[63]																																FC0h-FFCh

TBD

Table 685 - DC_CNAK_TRACE_TABLE Structure Fields

Offset	Bits	Name	Description	Access
00h-FFCh		dc_c-nak_trace_entry[0]		

Each DC_CNAK_TRACE_ENTRY looked as described in Table 686:

Table 686 - DC_CNAK_TRACE_ENTRY Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lid[0]																lid[1]																00h

Table 686 - DC_CNAK_TRACE_ENTRY Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lid[2]																lid[3]																04h
...																																...
lid[28]																lid[29]																38h
sequence_number																																3Ch

Table 687 - DC_CNAK_TRACE_ENTRY Structure Fields

Offset	Bits	Name	Description	Access
00h-38h	31:16	lid[30]	Fan tachometer index	Index
3Ch	31:0	sequence_number	Interrupt Enable. If the ie bit is set and the Tachometer reading is out of range as defined by tach_min and tach_max, the FERR# will be asserted.	RW

Sequence_number behavior:

- First Entry with sequence_number=0.
- sequence_number of each entry will be incremented by 64. i.e. second entry 64, third 128...
- sequence_number used to indicate if entry is valid or not.

17 Data Integrity

The device implements advanced techniques to assure data integrity for both hardware and software:

1. Hardware level - the device implements all mandatory and optional data integrity checks on its physical interfaces (network interface, PCIe interface). Hardware assures smooth recovery in an unlikely event of internal data corruption (SER or alike).
2. Software level - the device validates data integrity of memory-resident control objects and work queues, reducing probability of bogus I/O operation caused by software that strolled through control objects and corrupted them.
3. Application level - the device implements hand-over of different data integrity domains at application level while executing I/O operation. Application level data integrity is discussed in chapter [Section 17.3, “Application Data Integrity: Signature Handover Operations,”](#) on page 880.

17.1 Hardware-level Data Integrity

The device is designed to assure soft error SDC MTBF of 50,000 years. The device also implements end-to-end data protection on its data path.

Error recovery and correction error logic is designed to correct internal errors.

Atomic and other certain operations are instantiated multiple times in hardware and an error indication is set if there is no consensus between them.

Data integrity errors are detected and reported to the software as specified in [Table 688](#).

Table 688 - Data Integrity Detection and Reporting

Interface / Component	Method of Detection	Method of Reporting
PCI Express Interface	LCRC, ECRC and Advanced Error Reporting header.	Per PCI Express specification
Network Interface	ICRC, VCRC, Ethernet CRC, FC CRC	As specified in respective network standards. A packet that experienced CRC error is discarded and carefully counted in respective bin.
Internal operation errors	Majority vote	<i>Internal Error</i> counter bumped.
Flash Contents	16-bit CRC	The device boots in non-operational mode (EEPROM burner mode), or the device boots normally but fails upon the RUN_FW command and reports the problem.
SER Exposed Internal Logic (All Memory Arrays)	ECC, CRC	Respective counter bumped up

Table 688 - Data Integrity Detection and Reporting (Continued)

Interface / Component	Method of Detection	Method of Reporting
SER Exposed Internal Logic (internal HW registers)	ECC	Also known as FFSER. Causes immediate disabling of both PCI and IB interfaces. Reset is required in order to continue operation. [EMC] Residue is still left in health buffer, and can be read after reset. See Section 27.2, “Health Buffer,” on page 1049).

Data integrity checks/generation overlap between different domains (for example, “check integrity upon data receive” and “generate ECC before store”).

The PCI-Express bus supports data integrity checks. The data integrity checks on the PCI-Express interface log the address of the access that caused the failure. The address can then be reported to software and can be associated with a transaction by the application running on top of the HCA device. Subsequently graceful recovery can be done.

17.2 Software-level Data Integrity - Control Objects’ Consistency Checks

The device implements basic protection against buggy SW that can corrupt its control structures and objects.

17.2.1 Device Configuration and Control Communication

1. As a first step in protection, the device does not expose its register space for writing. The control communication with the device is implemented via a Command queue mechanism. See [“HCA Command Queue” on page 613](#) for more details. Software writes a control message to a Command queue in system memory.
2. Software seals the Command with an 8-bit XOR signature.
3. If the command includes an input mailbox, the mailbox is also protected by the command signature. This include all reserved fields of the mailbox.
4. Software informs the device that such command was posted by writing to the “command DoorBell vector” register in the device. (This is one of the rare writes that is NOT ignored by the device).
5. The device reads the command from the Command Queue, validates the signature, and will only proceed if the validation passed.
6. The device writes the output mailbox, updates the status of the command, regenerates the signature such that it will include the updated status and the output mailbox (input mailbox is not included).

17.2.2 Memory-resident Control Objects

Control structures required for the HCA operation (context tables, translation tables etc.) are kept in the host memory allocated for the device. This memory is called Interconnect Context Memory (ICM) and it is essentially a part of the HCA hardware. See [Section 6.1, “Interconnect Context](#)

[Memory](#),” on [page 239](#) for details. Host software should never access memory allocated for the HCA control objects.

Since the HCA cannot prevent corruption of its ICM memory located in host memory, it implements basic consistency checks.

Consistency checks are implemented by attaching a signature to every control object. This signature is generated by the HCA every time it writes an object to memory and this signature is validated every time it reads the object from memory. If the consistency check fails, the device treats this object as “not present” (invalid) and generates an event/interrupt to the host software, indicating address of memory location that was corrupted. If a context object corruption is detected during an I/O operation, it will complete same way as if the corrupted object was invalid or missing. Other I/O operations will not be affected and execution will proceed.

In order to generate the event notifying of the corruption of a control object, the HCA must read from ICM memory various control objects for the event queue designated for these reports. If one of these objects is corrupted, it is impossible for the HCA to generate the event. In this case, HCA will issue “CONTROL OBJECT CORRUPTION REPORT FAILURE” interrupt; status information with object address and object ID that caused this failure can be read by software from “control corruption” register.

We cannot guarantee that HCA can always handle ICM corruption gracefully. For instance, failure to post EQE reporting object corruption is considered to be a fatal error and HCA will halt execution - all committed data transfers (PCI and network) will be completed, but no new operation will start and the HCA will halt awaiting for hardware reset. The mean of this special interrupt (pin, MSIX vector etc.) is configured at HCA initialization step as described in [“Initialization and Teardown” on page 243](#).

17.2.3 Work Queues Elements Signature

Work Queue Elements (WQEs) undergo syntax/consistency checks as described in the [“Work Request \(WQE\) Formats” on page 310](#). These checks assure that operation specified in the WQE is consistent with queue this WQE is posted on. In addition to these syntax checks, the HCA can check a signature written on each WQE.

The WQE signature is generated by software posting the WQE. In addition to the WQE itself, the QP/SQ/RQ number of the queue the WQE is posted to, and the index of that particular WQE are considered in the signature. Send WQE signature is within the control segment. Receive WQE signature is in a different, additional segment.

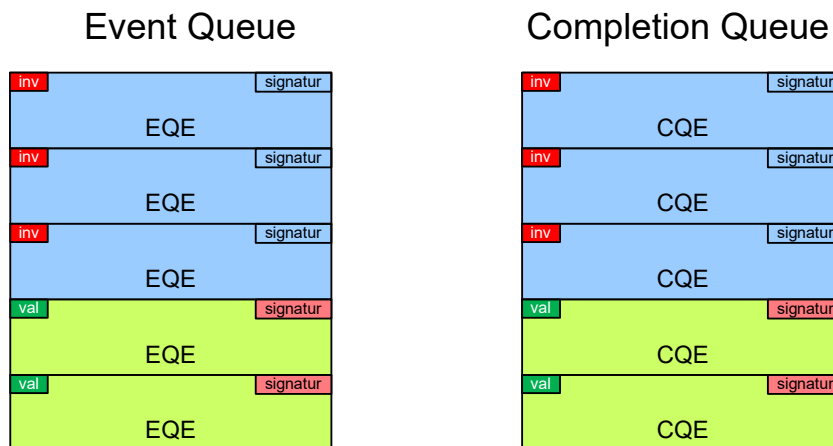
Signatures must be generated by the software posting Work Request, which consumes CPU cycles. However, in many cases posting Work Request is a performance-critical operation. The device provides an option to skip consistency checks, thereby avoiding this software overhead on post operation. Each QP/WQ can be independently configured to enforce WQE signature validation software. For QPs connected to an SRQ, configuration is per SRQ. See [“Queue Pair Context Format” on page 258](#) for more details about WQ configuration for WQE signature protection.

17.2.4 Completion and Event Queue Elements (CQEs and EQEs)

The HCA inserts a signature to Completion Queue Elements (CQEs) and Event Queue Elements (EQEs) when writing them. While polling the element from the queue, consumer can check the signature.

Figure 71 illustrates schematically CQ and EQ containing signature, showing valid and invalid entries in the queue. Detailed CQE format is discussed in [Section 8.19.1.1, “CQE Format,” on page 435](#) and EQE in [Section 8.20.2.2, “EQE Format,” on page 474](#).

Figure 71: Completion and Event Queues



Hardware always generates signatures for CQEs and EQEs. Validation of these signatures is under software control. Consumer can save CPU cycles by ignoring signature field, thereby accelerating CQ or EQ poll function calls.

17.3 Application Data Integrity: Signature Handover Operations

The device implements hardware offload of signature operations for user data. Signature handover is done by replacing one application signature with another one while executing I/O operation. An example of such a signature handover is sending a large data structure stored in multiple disk blocks of 512 bytes with associated DIF signature to destination expecting to receive this structure as a multiple of disk blocks of 1Kbyte with per-block DIF signature.

The device supports signature rules as specified below:

- No signature (raw data)
- T10 DIF - T10-compliant Date Integrity Field
- [XOR-8 - bit-wise XOR operation on the entire data structure](#)
- XOR-32 - bit-wise XOR operation on the entire data structure
- CRC-32 - 32-bit CRC
- T10DIF_CS. - T10-compliant Date Integrity Field, with Checksum guard instead of CRC
- CRC32C Signature Rule
- CRC64_XP10 Signature Rule
-

The device supports signature handover operations as specified below:

- Append signature
- Validate signature
- Validate and replace signature
- Validate and strip signature
- **Validate signature and replace LBA (T10 DIF only)**

The device supports block-level signature operation (for example T10 DIF block-level protection) - in which case signature handover is executed on every data block as an independent operation. In addition, the device supports transaction-level signature operation - where transaction is a data structure that can contain multiple data blocks with signature-level protection as well as opaque data - and entire data structure is protected by its own signature of one of the supported types.

17.3.1 Signature Rules and Formats

This section lists signature rules and signature formats supported by the device.

17.3.1.1 T10 DIF Signature Rule

T10DIF rule is defined in the T10 specifications (T10 SBC-3) for block-level data protection. The size of data block protected by T10 DIF must be modulo 8bytes as required in the T10 DIF specifications. Note that when setting the initial LBA value (via SET_PSV), it should be the value of the first block to be transmitted. After the transaction if the LBA is retrieved (via GET_PSV), it will be the value of the last block transmitted + 1.

17.3.1.2 XOR8 Signature Rule

XOR8 signature is created by executing a bitwise XOR operation of all bytes of the data structure. Integrity check passes if bitwise XOR of all bytes (including a signature) yields zero result.

The size of data block protected by XOR8 is not constrained.

17.3.1.3 XOR32 Signature Rule

XOR32 signature is created by executing a bitwise XOR operation of all DWORDs of the data structure. Integrity check passes if bitwise XOR of all DWORDs (including a signature) yields zero result.

The size of data block protected by XOR32 should be modulo 4 bytes. If not, HW calculate the

17.3.1.4 CRC32 Signature Rule

CRC32 signature is created by calculating a 32-bit CRC defined in Fibre Channel Physical and Signaling Interface (FC-PH), ANSI X3.230:1994. Integrity check passes if signature calculated on the data matches the CRC32 value.

The size of data block protected by CRC32 must be module of 4 bytes.

17.3.1.5 T10DIF_CS Signature Rule

Same format and rules as T110DIF, but use checksum instead of CRC in the guard field.
Integrity rules are identical to IP checksum rules

17.3.1.6 CRC32C Signature Rule

CRC32C signature is created by calculating a 32-bit CRC called the Castagnoli CRC, defined in the Internet Small Computer Systems Interface (iSCSI) rfc3720.

Integrity check passes if signature calculated on the data matches the CRC32c value.

The size of data block protected by CRC32c must be module of 4 bytes.

17.3.1.7 CRC64_XP10 Signature Rule

CRC64_XP10 signature is created by calculating a 64-bit CRC defined in Microsoft XP10 compression standard.

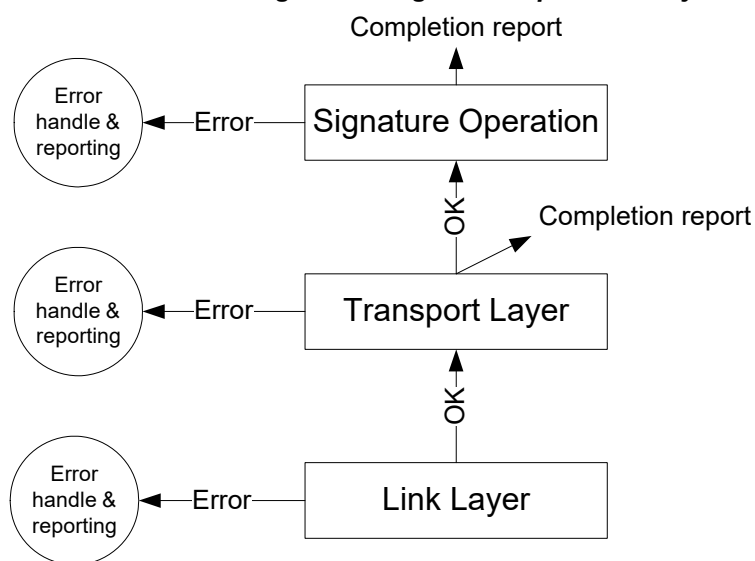
Integrity check passes if signature calculated on the data matches the CRC64 value.

The size of data block protected by CRC64_XP10 must be module of 8 bytes.

17.3.2 Signature Handover Operation Layer

Signature handover operation is done in a strict layered manner as shown in [Figure 72](#). Thus, completion of a signature operation (including error report) is reported independent of transport completions and a signature operation error does not affect the status of the transport connection.

Figure 72: Signature Operation Layers



17.3.3 Variable, Repeating, and Transaction Signatures

Three types of signatures are implemented.

Variable block signatures are non repeating and can therefore cover data of any size. The initialization, check, strip, or addition of any variable block signature must be explicitly specified.

Repeating block signatures are repetitive and all blocks must be of equal size. Repeating block signatures are automatically initialized, checked/stripped/added at the appropriate time.

Transaction signatures are used in addition to variable and/or repeating block signatures. A transaction signature can cover many block signatures to protect an entire transaction. There is a limited subset of signature types supported for transaction signatures. Only XOR-8, XOR-32 and CRC-32 signature types can be used for transaction signatures.

17.3.4 Signature Operation Arguments

A signature operation is executed between two *Signature Domains* - *Memory Signature Domain* and *Wire Signature Domain*. The data structure in either signature domain is constructed of data blocks of potentially different sizes and potentially different signature rules. In addition, the data structure can contain *opaque* data chunks of arbitrary size that are not protected by a block signature.

Figure 73: Signature Diagram Key

Signature example diagrams generally show the memory domain as the source and the wire domain as the destination (RDMA write/SEND), but either domain can be used as the source or destination.

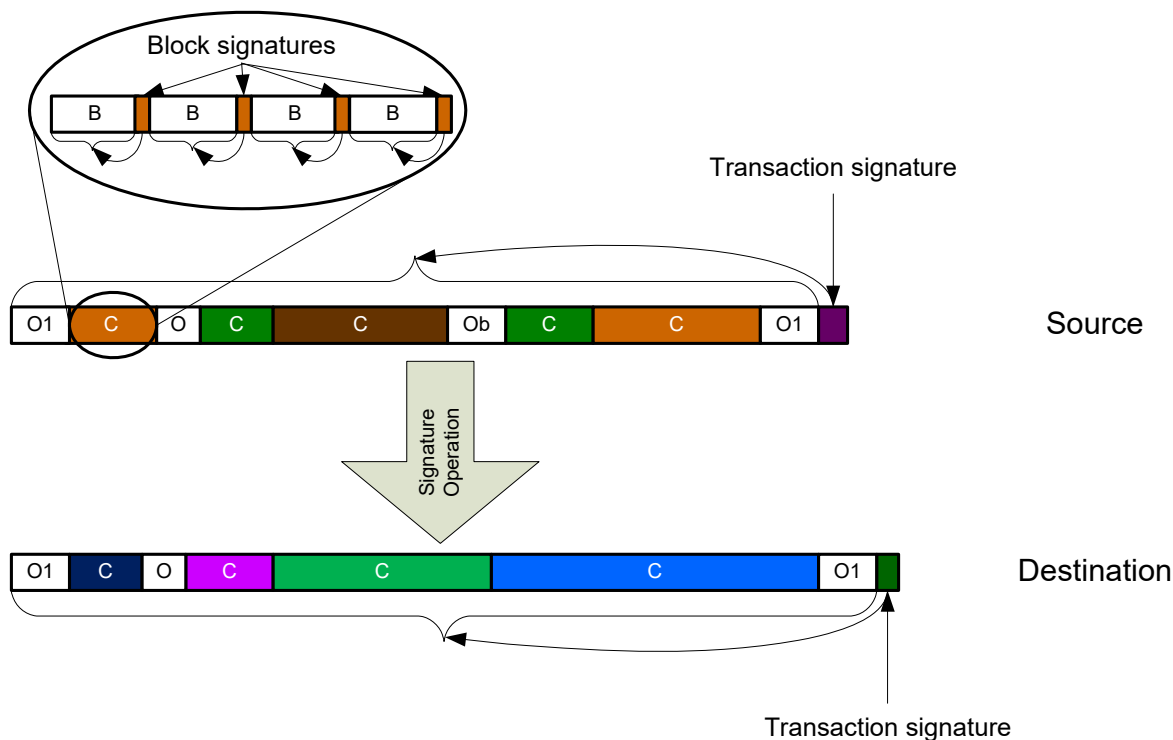
The following abbreviations are used for all signature diagrams

C: Chunk of data blocks that share the same signature properties (type/size)
 B: One data block covered by a signature
 S: Signature
 T: Transaction signature
 O: Opaque (non signature protected) data
 m: Memory domain
 w: Wire domain

The most general case of a signature operation transaction is shown in [Figure 74](#). Data chunks marked “C” include signature-protected data blocks (structure of such a chunk is discussed in “[Signature Check Without Data Transfer and Data Padding](#)” on page 885). Different colors designate different signature rules and/or different size of the data block. White chunks marked “O” designate opaque data.

The coverage of block-level signature and transaction signature is also shown in the figure at high level. A detailed picture is presented and discussed later in the document.

Figure 74: Signature Operation

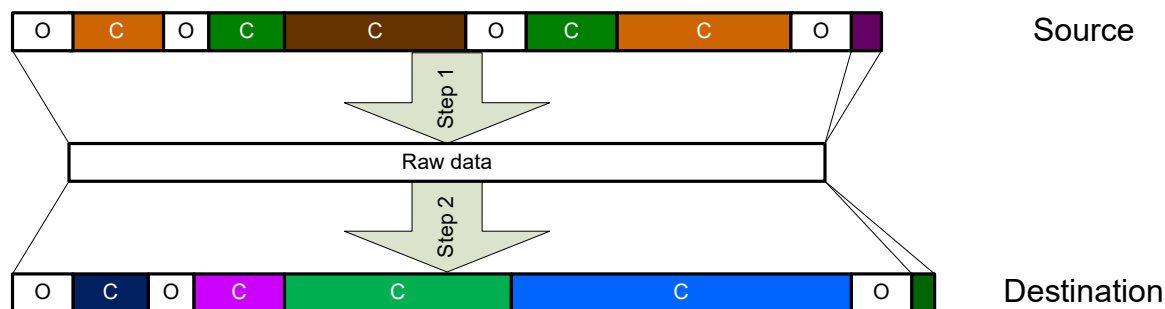


Each domain can be either the source or destination. Signature operation is supported for send/receive as well as RDMA (read and write) operations on either requester or responder side.

17.3.5 Signature Handover Operation Steps

Signature handover operation should be viewed as two independent steps. The first step is to process the source data, verifying and stripping all signatures. This results in a raw data stream that is fed to the second step of the signature operation, calculating and appending all signatures for the output stream. These steps are illustrated in [Figure 75](#).

Figure 75: Signature Handover Steps



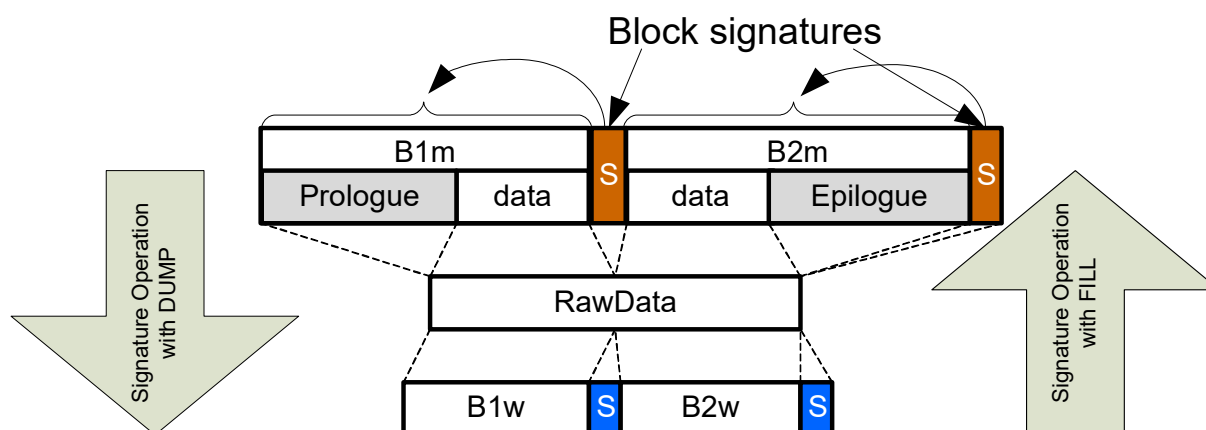
Although two steps shown in [Figure 75](#) are executed in pipeline (that is, signature-free data is not stored prior to generating output data stream), it is important to view these two steps as independent to avoid confusion.

17.3.6 Signature Check Without Data Transfer and Data Padding

It may be desired to transfer only a piece of the data protected by a signature, yet still use the signature to validate that data. To accomplish this, each signature protected chunk in the memory domain can have “prologue” and/or “epilogue” bytes. When the memory domain is the source (for example write operation), these bytes will be used in calculating the signature, but will not be transmitted to the wire (dumped). When the memory domain is the destination (for example read operation), the prologue and epilogue bytes will be zero filled. This is how to zero fill data in memory. The creative software engineer could even use this method to get the HCA to initialize large sections of memory.

The same operations can be performed in the wire domain albeit through a different mechanism. A special memory key is provided that can be thought of as a bit bucket or zero generator (or / dev/null). When the memory domain is the destination (for example read operation), this key can be used in the scatter entry to dump the bytes that are not desired to be written to memory. The bytes are dumped after they have been used in calculating any applicable signature(s). When the wire domain is the destination (for example write operation), using this key in the gather entry will tirelessly generate as many zeros as required. The mischievous software engineer could even use this method to get the HCA to initialize large sections of someone else’s memory. When using the special key, the data is dumped or the zeros are generated without generating any traffic on the PCIe bus.

Figure 76: Partial Data Transfer



In the above diagram, both the prologue (marked as Wh), epilogue (marked as Wt) and data sections are considered part of each block, and are included in the memory signature calculation. However, only the “data” part will be transferred to/from the wire. Prologue and epilogue will be dumped or zero filled, depending on direction.

17.3.7 Padding

When the pad size is less than 4 bytes, and is immediately preceding the signature, it is recommended to avoid using prologue and epilogue. Instead, one can use the “padding” feature.

With the padding feature, the user specifies the number of bytes that should be stripped (source domain) from the end of the data and should be added (with zero value) at the end of the data before generating the destination signature. These bytes are covered by the signature.

Note that pad bytes are considered attached to the signature they precede, they can not stand on their own.

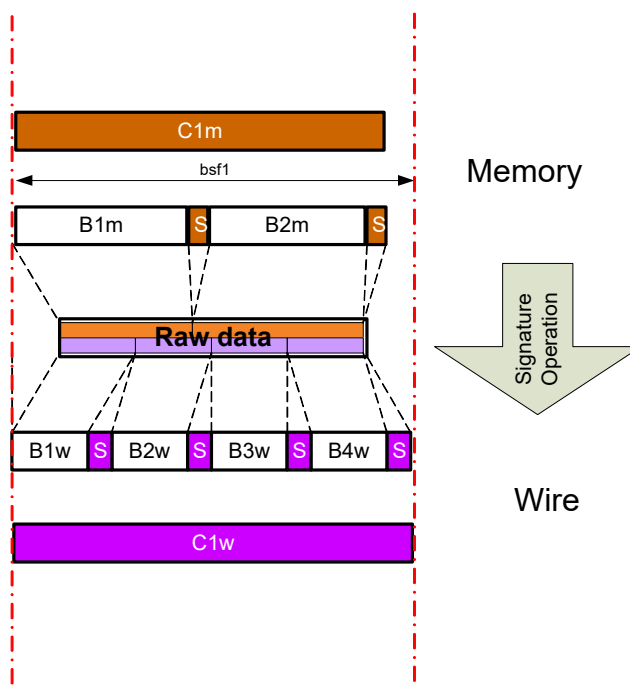
17.3.8 Signature Operations’ Parameters

Signature operation is controlled by two types of HCA control objects associated with data transfer - *Byte Stream Format(s)* (BSF) and *Persistent Signature Variable(s)* (PSV). Each signature handover operation can use one or more *BSFs* and one or more *PSVs*.

Each BSF covers a specified number of raw data bytes (signatures are not included in the byte count). It defines all of the signature attributes of both the memory domain and the wire domain for these bytes. See [Table 695, “Byte Stream Format \(BSF\) Layout,” on page 902](#).

[Figure 77, “A Simple BSF”](#) shows a simple BSF defining 2 repeating blocks in the memory domain and 4 repeating blocks on the wire domain.

Figure 77: A Simple BSF

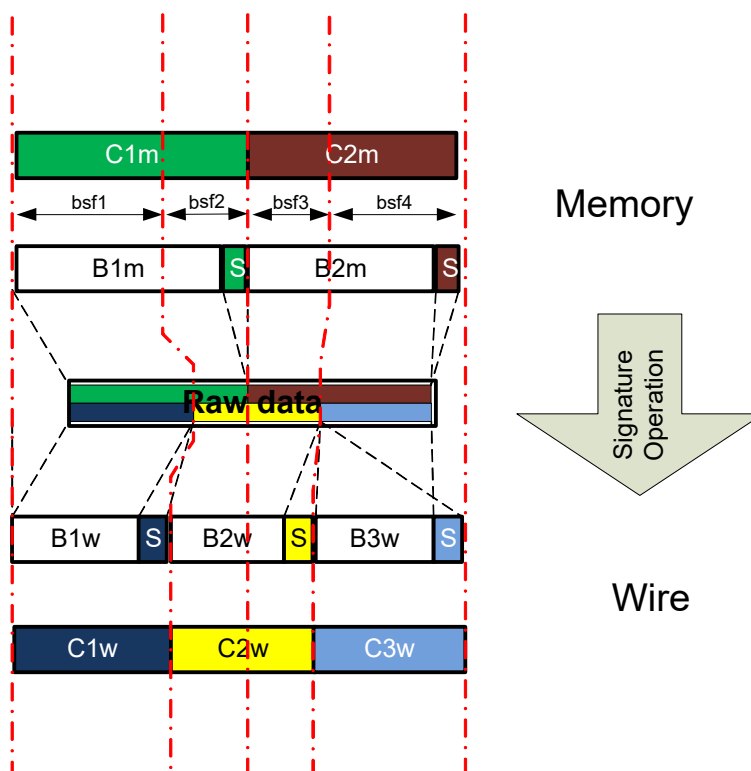


A single BSF will be sufficient to carry out most signature operations. More complex operations are possible, and will require multiple BSFs. A new BSF will be required for each of the following cases on either the wire or memory domains:

- Each block of variable block signature type
- Each time a repeating block signature type changes
- Each time a repeating block signature needs to change seed values (that is, LBA change for T10 DIF)
- Each time a new prologue/epilogue area is needed
- Each time a new transaction signature is needed

Figure 78, “Variable Blocks to Variable Blocks BSFs” illustrates a more complex set of BSFs that would be used when the memory and wire domains contain incongruent variable block signatures.

Figure 78: Variable Blocks to Variable Blocks BSFs



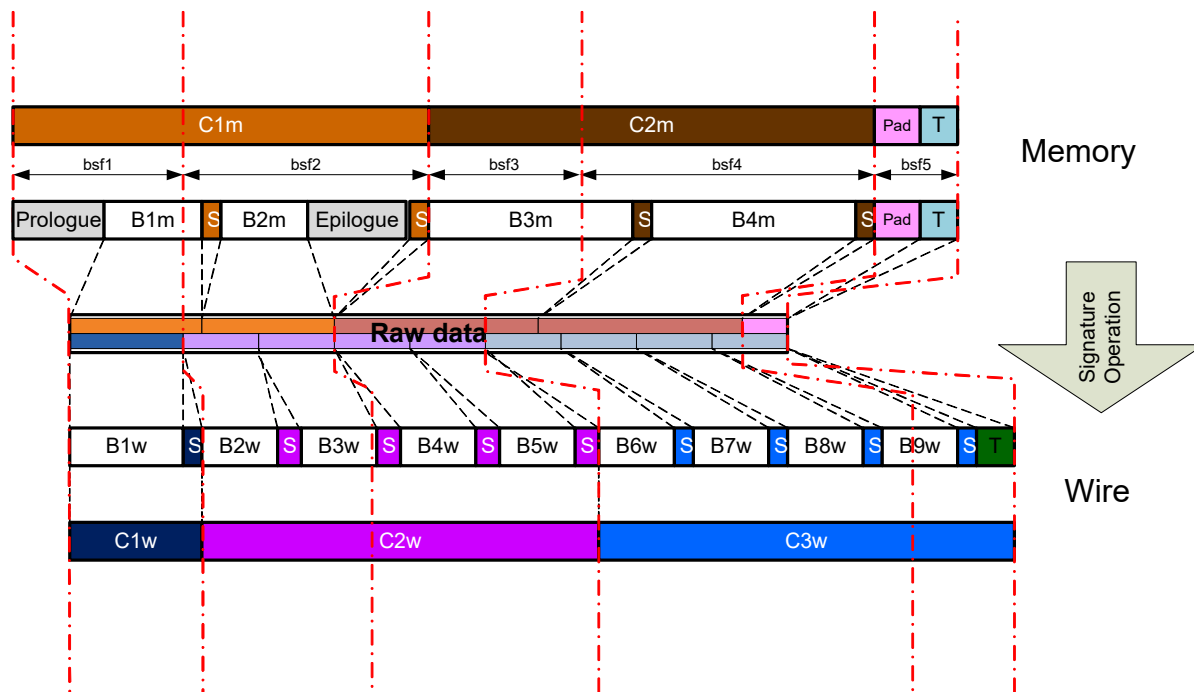
The above figure will also require correct coding of the *init* and *check_gen* bits of the BSF.

Table 689 - Init and Check_gen Example

BSF#	m_init	w_init	m_check_gen	w_check_gen
bsf1	1	1	0	1
bsf2	0	1	1	0
bsf3	1	0	0	1
bsf4	0	1	1	1

BSF configuration can get quite complex as illustrated in [Figure 79, “Complex BSF Configuration”](#).

Figure 79: Complex BSF Configuration



BSF1 covers data fed from C1m source chunk and used to generate C1w data chunk on the wire. Note that part of the data covered by this BSF must be dumped prior to transferring the data to destination domain. This would be accomplished by defining a prologue for BSF1.

BSF2 covers remainder of the C1m source chunk used to generate the first part of the C2w chunk. Note that part of this data covered by this BSF must be dumped before transfer of this data chunk is completed. This would be accomplished by defining an epilogue for BSF2. Although it may appear so, BSF2 does not describe the entire data part of B3w block. Several data bytes of this block are described by BSF3.

BSF3 covers data fed from C2m source chunk and used to generate the remainder of C2w chunk.

BSF4 covers remainder of data fed from C2m source chunk and used to generate first part of C3w chunk.

BSF5 covers padding that is added to source data to fit operand size required to generate C3d chunk. Note that the “Pad” in this example is actually transmitted on the wire (unlike the padding feature described in [“Signature Check Without Data Transfer and Data Padding”](#) on page 885 or [“Padding”](#) on page 886). This segment is also the last one in the transaction, so it must explicitly state to generate/check transaction signatures on the memory and wire domains. The transaction signature will be read and verified at this stage and output transaction signature will be generated.

BSFs are associated with memory keys via the UMR operation. See [“Signature Operations Parameter Registration”](#) on page 890 for more details.

A PSV is used to contain a calculated signature value for a single signature along with some meta-data, such as error flags and status flags. See [Table 690, “Persistent Signature Variable \(PSV\) Layout,” on page 899](#).

Each BSF is associated with up to 4 PSVs - two for the memory domain (block/transaction) and two for the wire domain (block/transaction). Software creates PSVs via the CREATE_PSV command (“CREATE_PSV” on [page 1876](#)) and uses the index returned by that command to associate a PSV with a BSF.

1. Prior to executing signature handover operation, PSVs must be initialized. There are 3 ways a PSV can be initialized (for example, post a SET_PSV WQE).
2. For variable block signatures, a flag can be set in the BSF that indicates that the PSV be initialized to a default value before the first byte of data of the BSF is transferred.
3. For repeating block signatures, appropriate portions of the PSV are automatically initialized immediately before the first byte of a block is transferred.

Note that this initialization refers only to the transient signature value (for example guard field of T10 DIF). All other fields in PSV may only be initialized by the SET_PSV WQE (option 1).

17.3.8.1 BSF Configuration Notes

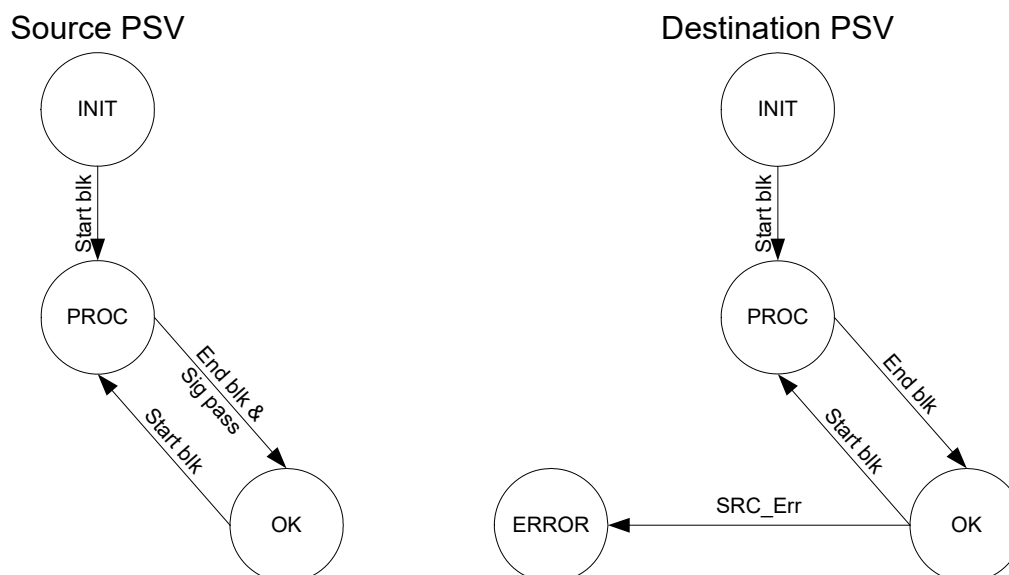
- A BSF can describe a single signature type in each of the domains: source block, source transaction, destination block and destination transaction. If more than one type is required in either of the domain, multiple BSFs are required.
- BSF describing a repeating block signature type cannot describe a partial signature zone, nor can it describe a data block without its signature. The signature is always attached to the last byte of the data block.
- BSF describing a variable block signature type with padding cannot describe the pad bytes without a generated/checked signature. The pad bytes are considered attached to the signature zone.
- The last BSF of a signature handover operation must describe at least 1 byte of data or signature in the wire domain, otherwise it will never be accessed.
- BSF same_block_structure should only be set if the memory and wire have identical structure. Prologue and epilogue sections are considered as breaking the identity.

17.3.8.2 PSV Status Transitions Heuristics

PSV status is initialized in the INIT state. HW will check the source signature versus the one it calculated after last byte of the source’s data block is processed and will then update the signature state. Once set to ERROR, PSV state can only be changed by re-initializing the PSV via the SET_PSV WQE.

[Figure 80](#) shows PSV status state transitions.

Figure 80: PSV State Transitions

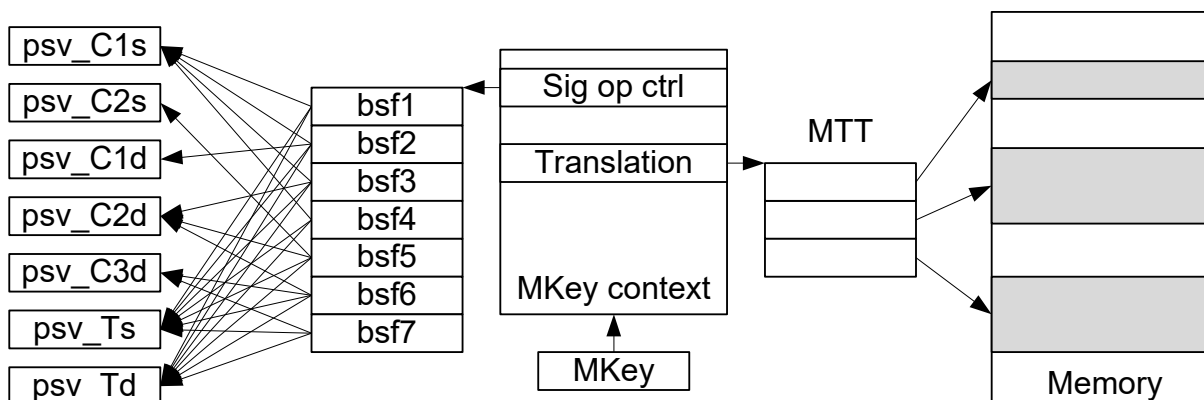


INIT, **ERROR** and **OK** states indicate an *idle* state of the PSV. After completion of a signature operation, all associated PSVs should be in one of the *idle* states. If one of the PSVs associated with I/O operation is in **PROC** state after I/O operation was completed, it means there is most likely a SW bug in operation setup and the programmer is advised to fix it.

17.3.9 Signature Operations Parameter Registration

Signature operations are implemented through the memory key mechanism. See [“Address Translation and Protection” on page 622](#) for a detailed discussion of memory keys. Upon creation, a flag in the memory key will indicate that this memory key is associated with signature operations and there will also be a number of BSFs in the BSF table associated with this memory key. Each memory key that is associated with signature operations usually uses indirect memory regions (see [“Address Translation Indirection” on page 624](#)). This creates a virtually contiguous memory space for the signature operations, but allows for the actual data to be scattered anywhere in memory. The UMR WQE (see [“User-mode Memory Registration \(UMR\) WQE Format” on page 322](#)) is used to fill in the BSFs and KLMs for the signature MKey at run time. Note that the KLMs or MTTs of a signature Mkey are specified in the memory domain. This means that KLMs and MTTs include and must account for any signatures that exist in the memory domain.

Figure 81: Signature Operand Memory Mapping



Hardware does not prevent associating multiple MKeys to map memory-resident signature operand and/or associating multiple MKeys with the same BSF.

17.3.10 Memory Access - Address and Length Calculation

Memory address and length calculations for non-signature operations:

WQEs and RDMA operations both use memory references in a WQE in the form of an MKey, virtual address and length. Hardware validates that the memory reference's virtual address fits within the region specified by the MKey, and calculates the offset of virtual address by subtracting the virtual address from the Mkey's starting address.

For indirect Mkeys, this offset will be used to select the appropriate KLM. If there are any KLMs before the selected KLM, the sum total of all of the prior KLM sizes is deducted from the offset, and this new offset is used to calculate the address to be used with the selected KLM. The process then begins again with this new Mkey and address.

For direct MKeys, this offset is used to look up the appropriate Memory Translation Table (MTT) entry to translate the given virtual address to a physical address.

The length of the transaction will continue to be the length argument as specified in the memory reference.

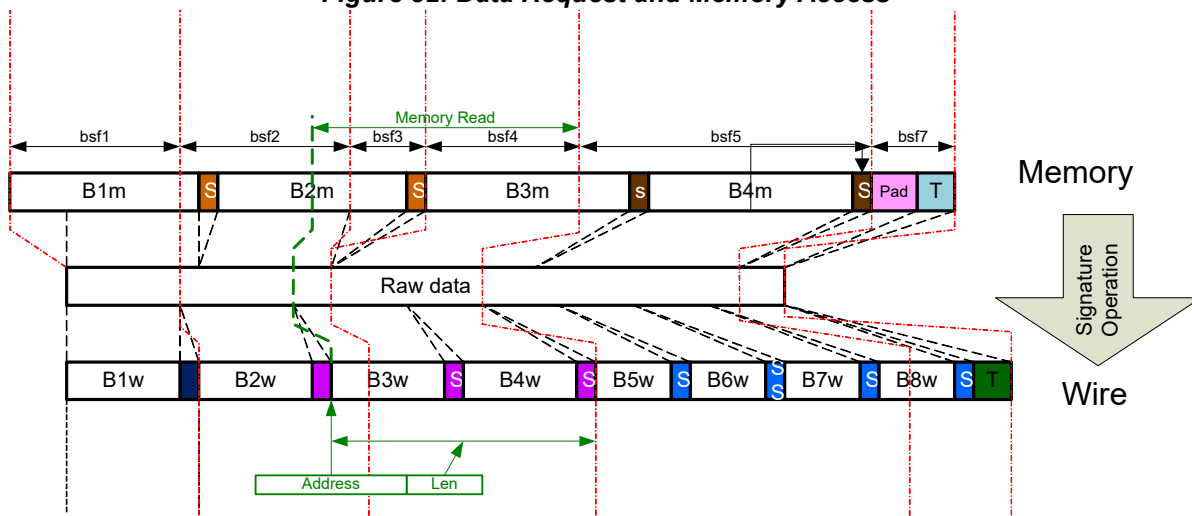
How signature operations changes this procedure:

If the MKey used by the memory reference is associated with a signature handover operation, the virtual address given in the memory reference is in the wire protection domain, and indicates the offset inside the signature transaction. This means that all memory references include and must account for any signatures that exist in the wire protection domain.

Since we are adding and removing signatures, it is likely that the offset of the virtual address of the memory reference does not equal the offset of the virtual address of the MKey. Similarly the length given in the reference may not equal the length that needs to be read from memory.

When accessing a signature type MKey, the HCA will calculate the memory domain offset from the wire domain offset by walking the BSFs. Similarly, once the starting address is calculated, it will adjust the length of the request as necessary.

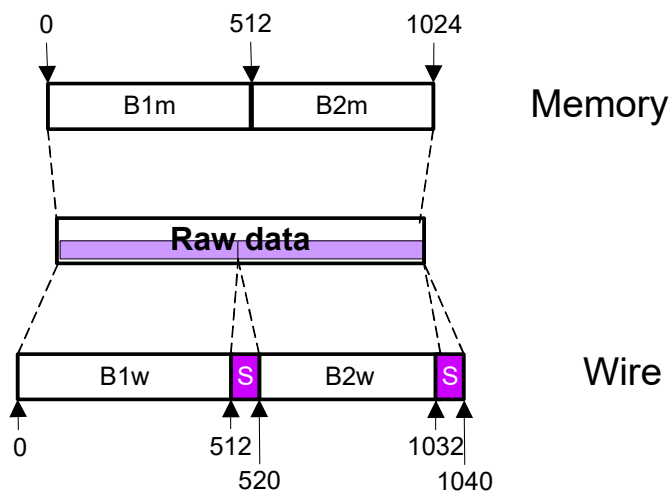
Figure 82: Data Request and Memory Access



For example, as illustrated in Figure 82, neither the offset from the beginning of memory buffer nor the number of bytes read from memory match the address and len specified by the original request. Rather, both these parameters were calculated by looking up the BSFs associated with the MKey used in the original memory request

.Memory requests generated by “wireless” WQEs follow same address calculation rules as specified above.

Figure 83: Address and Length Adjustment Example



In this figure, we show an example of how wire to memory address adjustment may take place. In this example, 2 512 byte blocks without signature exist in memory. We wish to add an 8 byte DIF signature on the wire to each of these blocks. Also for this example, let’s assume we will use a separate WQE for each block.

For the first block, the WQE would specify address 0 and a length of 520 bytes. Remember that WQE’s are always specified in the wire domain! The starting address would not change, since

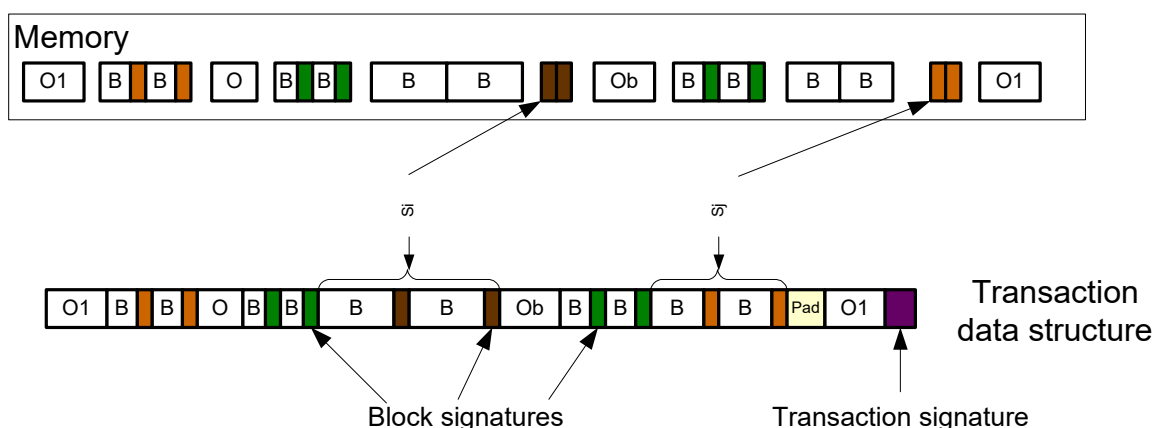
address 0 is equivalent in both memory and wire domains. The length, however, needs to be adjusted since there is 8 bytes of signature on the wire, but none in memory. The number of bytes read from memory must be reduced by 8 to reflect this. Therefore the length to read from memory is adjusted to 512.

Again, remember that the WQE is always specified in the wire domain, so it would be coded as address 520 and a length of 520 bytes. HW will look at the BSFs and see that address 520 on the wire is actually address 512 in memory, so it will adjust the starting address to 512. Then, in a similar fashion to the first block, the number of bytes to read will be adjusted to 512.

17.3.11 Data and Signature Collocation

The signature handoff mechanism requires that each signature be located immediately after the data it protects. If the signatures are not collocated with the data, perhaps in a separate “signature block”, UMR can be used to create a virtually contiguous memory domain, inserting each signature after the block it protects See [Figure 84](#).

Figure 84: Signature Non-collocated



In this example for memory-resident data structure S_i and S_j signature(s) are located in separate (“meta-data”) buffer.

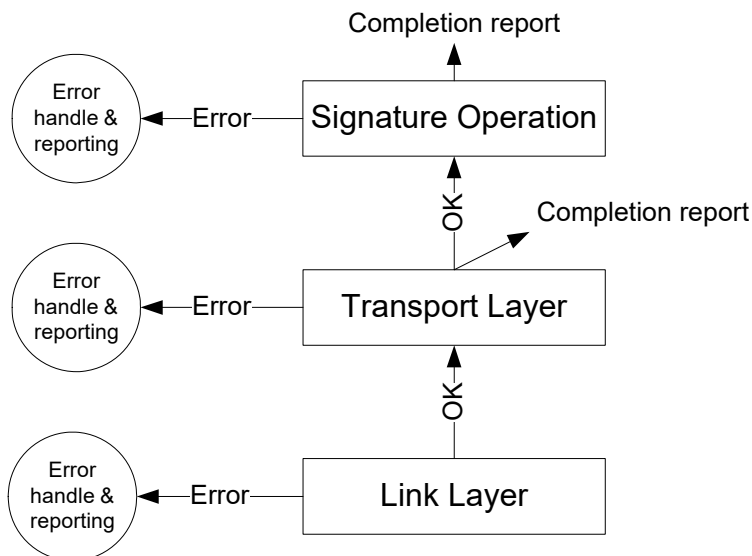
17.3.12 Data Structure Size and Address Alignment

The device supports block/transaction size as small as 1 byte. HW imposes no practical limitations on block/transaction size or on memory address alignment¹.

The only limitation exists is that the amount of data and signatures generated by MTU bytes in a single BSF. When doing the signature handover on the send side MTU bytes on the wire should not generate more than 20KB in the memory. When doing the signature handover on the receive side MTU bytes on the wire should not generate more than 10KB in the memory.

1. Performance may be compromised for weird sizes and address alignments. This will be part of performance guidelines.

Figure 86: Signature Operation Layers



A signature transaction can be implemented with multiple I/O operations (send/receive or RDMA), possibly even on multiple QPs. Programmer must assure that only one QP executes an I/O operation associated with a given signature transaction at any given time. If all I/O operations are executed on same QP, this requirement is satisfied by construction.

If a signature transaction spans across multiple queues, execution of I/O operations on different queues must be explicitly serialized. Serialization can be done either by software or by deploying network operations' synchronization mechanism supported by the HCA as explained in [Section 20.1, "Cross-channel I/O Operations," on page 980](#).

Execution of signature handover operation involves three explicit steps:

1. Initialize the control structures (BSFs, PSVs and MKey) to be used by HW to identify data transfers involved in a particular signature operation.
2. Transfer data subject to signature operation through the device.
3. Obtain result of signature handover operation

17.3.14.1 Signature Operation Data Transfer Work Requests

Signature operation is executed while transferring the data through the device. Amount of data received from the source (either memory or wire domain) may differ a lot from the amount of data generated on the destination - depending on signature addition or stripping and also if prologues or epilogues (dump and fill) are used.

17.3.14.2 Signature Handover Completion Report

Hardware does not maintain the notion of the beginning or end of a signature transaction. The signature transaction is completed when a completion is reported for the last message of the signature transaction. This completion report is a network transport layer report (for example ACK received from remote peer for send or all data received on responder) and does NOT include indication of the success or failure of a signature operation.

Hardware can be configured to generate s SIGERR_CQE ([Table 161, “SIGERR CQE Entry Format Layout,” on page 458](#)) for Mkeys armed to generate SIGERR_CQE upon encountering a signature error. Hardware ensures that SIGERR_CQE is generated before a signature transaction is finished.

For signature handover on the requester side, and only when `HCA_CAP.drain_sigerr==1` (See [Section 29.3.4, on page 1667](#)), HW can also be configured to move the QP to SQD state upon encountering a signature error and generate a `Send_Queue_Drained` Event to inform about the new state.

To enable this feature, software must:

- Set `QP.drain_sigerr` to 1 when creating the QP (See [Section 8.3.5, “Queue Pair Context Format,” on page 258](#)).
- Register to `Send_Queue_Drained` Event (See [Section 8.20.6, “Event Queue Mapping,” on page 476](#)).

After reaching the SQD Event, SW should perform `QUERY_QP` to get last WQE reached in sq, then can use the `SQD2RTS` command to return to the RTS state (See [Section 29.10.12, “SQD2RTS_QP – SQD to RTS QP,” on page 1929](#)).

Another way to determine the success or failure of a signature operation, the user can execute the `CHECK_PSV` Work Request. If the user needs further additional information, like the final signature value (that is, for seeding the next operation), it can execute the `GET_PSV` Work Request. More debug information (extended PSV) can be obtained by using the `QUERY_PSV` command.

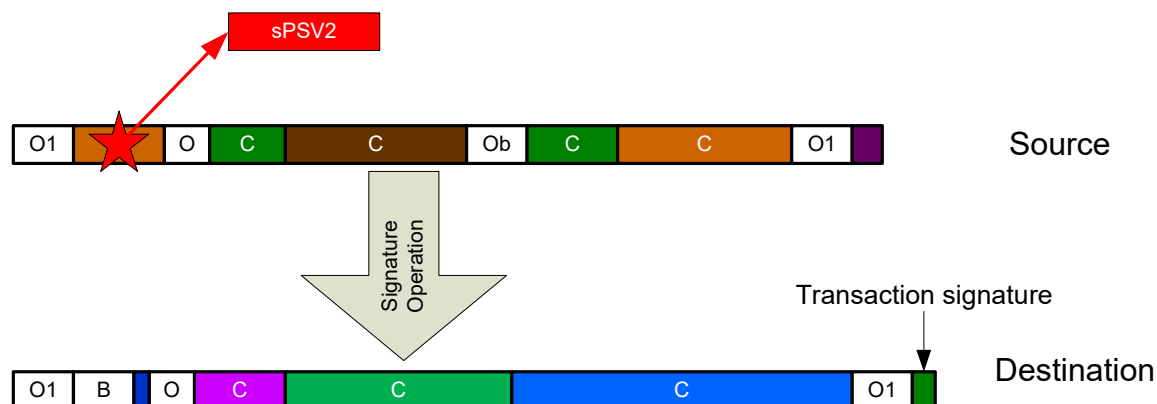
17.3.15 Signature Operation Error Reporting

As explained in Signature Handover Completion Report ([Section 17.3.14.2, “Signature Handover Completion Report,” on page 895](#)), if signature error is encountered while executing signature operation, Hardware can be configured to generate s SIGERR_CQE for Mkeys armed to generate this type of CQE ([Table 161, “SIGERR CQE Entry Format Layout,” on page 458](#)). SIGERR_CQE will include all information needed about the error occurred.

This sad event will also be logged in respective source PSV (sPSV) by setting its status to error state.

This is illustrated on [Figure 87](#), where the red star indicates the input data structure that experienced a block signature error.

Figure 87: Signature Operation With Error

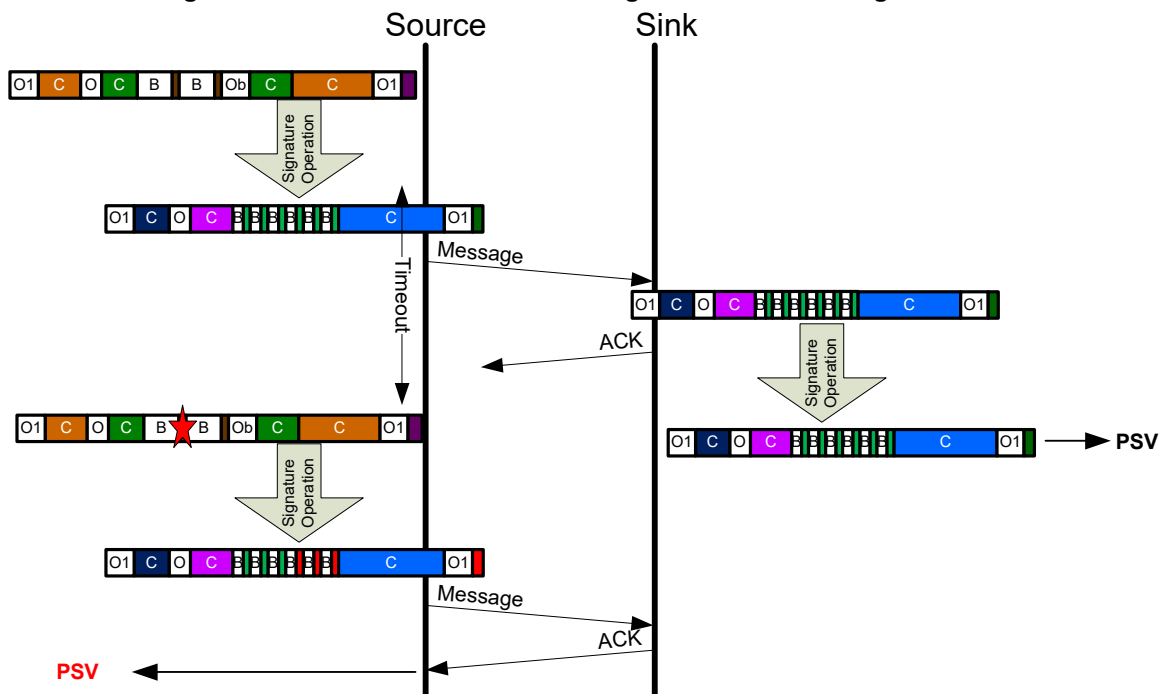


Additional debug information is logged in the PSV that detected an error (see Table 690, "Persistent Signature Variable (PSV) Layout").

17.3.15.1 Signature Handover Status Synchronization

If signature handover operation is executed both on data source and data destination, transport error along with data corruption on the source can result in disagreement of two sides of the connection with respect to data integrity as shown in Figure 88.

Figure 88: Source and Destination Signature Check Disagreement



Resolution of such a disagreement is beyond the scope of this document.

17.3.15.2 Pipelined Status Notifications

Certain storage protocols (that is, FC target, SRP target etc) indicate data transfer completion of an I/O read operation by sending a control message following the data. If a signature handover operation is executed while sending the data, a completion message is conditioned on error-free signature operation.

The device enables conditional flush of send operations based on memory-resident conditions, which enables pipelined posting of completion messages behind data messages. Condition execution of these completion messages based on signature handover as a result of previous operations is possible. [TBD add link to more details](#)

17.4 Signature Handover Programming

17.4.1 Signature Handover Steps

In order to execute signature handover, programmer must execute the following steps:

1. Allocate memory key to be used for the operation out of a pre-configured pool of signature enabled memory keys.
once memory key is created, a counter of expected `sigerr_cqe_count` should be created with init value of 1. This counter should be incremented by 1 each time `sigerr_cqe` is received on this memory key.
2. Configure the memory key via the UMR command:
 - a. Create a gather list describing the data structure layout in memory and program the memory key's KLM list. This creates the virtually contiguous memory necessary for signature operation.
 - b. Ensure that data structure begin from the beginning of a protected block and Must not begin from the middle, i.e cannot cover just part of a protected block.
 - c. Describe the signature properties of the memory and wire domains via BSFs.
 - d. Set `expected_sigerr_count` in Mkey Context to the LSB of the counter that counts expected `sigerr_cqes` ([Table 362, "MKey Context Format," on page 629](#)), otherwise,
3. Initiate signature parameters by posting SET_PSV Work Requests on the same queue (see ["SET_PSV Work Request" on page 913](#)). This step is not required if initial signature value is its default value and its the first use of this psv.
4. Post one or more Work Requests (send, RDMA-write, RDMA-read, or receive) that use the MKey from prior steps to initiate the signature operation:
 - a. Work Requests are specified in the wire domain, so the total byte count in the gather list of these I/O operations must specify the number of bytes to be transmitted on the wire.
also the address in the gather list define the offset from the region and not the VA of the data in memory. i.e first byte in the memory region will be transmitted with VA=0.
 - b. Each Work Request must begin from the beginning of protected block in both wire and memory domains. i.e , cannot post work request that transmit data from middle of protected wire or memory blocks.
 - c. Work Requests must be posted in the same order of the data structure they process.
5. The device generates SIGERR_CQE before the transaction finished (See [Table 161, "SIGERR CQE Entry Format Layout," on page 458](#)). This cqe is generated for the first signature error the device finds, and it is the user's responsibility to increment the counter that counts expected

sigerr_cqes and set its LSB in the expected_sigerr_count field in the Mkey Context for future use of this mkey, (See Table 362, “MKey Context Format,” on page 629). Post a CHECK_PSV Work Request. This Work Request automatically performs the signature check after the signature operation is completed.

6. If the final signature is needed, use the GET_PSV WQE to obtain it.
7. If a signature error was reported by the CHECK_PSV, use the QUERY_PSV command to obtain more information about this error.

Note that these steps describe executing all signature operations on the local HCA. The target has no concept of signature operations. It is theoretically possible for signature handover operations to be executed on the target as well. RDMA-read case is particularly tricky.

17.4.2 Signature Handover Control Structures

17.4.2.1 Persistent Signature Value (PSV)

Persistent Signature Variable is created by using the CREATE_PSV command (see “CREATE_PSV” on page 1876). PSV is initialized by passing the structure in Table 690, “Persistent Signature Variable (PSV) Layout” with a SET_PSV Work Request on a Send Queue.

Table 690 - Persistent Signature Variable (PSV) Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset	Base
1	0	9	8					pd																00h								
syndrome										reserved status																04h						
transient signature																08h																
reference tag																0Ch																
																10h	Extended															
signature error log: Mkey																14h																
signature error log: VA_h																18h																
signature error log: VA_l																1Ch																

Table 691 - Transient Signature Layout - For T10 - DIF, T10 - DIF CSUM

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	11 0	1 0	9	8	7	6	5	4	3	2	1	0	Offset 00h
block guard (CRC/CSUM)																application tag																

Table 692 - Transient Signature Layout - For XOR32, CRC32 signature

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2			0												00h
XOR/CRC																																		

Table 693 - Transient Signature Layout - For XOR8 signature

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2														00h
XOR8																																	

Table 694 specifies fields of the signature context.

Table 694 - Persistent Signature Variable (PSV) Fields

	Bits	Name	Description
☐	24	PSV handle / PD	24 bits PSV index
☐	1	valid	1 for valid PSV
☐		heart bit	
Base			
00h	23: 0	pd	Protection domain - reserved for SET_PSV WQEs
04h	31: 16	syndrome	Syndrome is a bitmask, in which each bit has the following meaning: 0 - alignment mismatch. Signature zone encountered after an unexpected number of bytes (according to signature type). This can often indicate a programming error, This bit is sticky. Once set, only SET_PSV can clear it. bits 1-6: reserved 6 - no_data_bit_is_set: all data bits are '0'. Valid for all signature types 7 - any_data_bit_is_set: not all data bits are '0'. Valid for all signature types 8 - middle_of_signature: indicates that the last byte calculated was a signature byte, but it was not the last byte of the signature 9 - last_byte_was_data: indicates that the last byte calculated was a data byte 10 - initialized: set on SET_PSV, cleared once the PSV is used 11 - reference_tag_error: mismatch was detected in the reference tag for DIF. This bit is sticky. Once set, only SET_PSV can clear it. 12 - application_tag_error: mismatch was detected in the application tag for DIF. This bit is sticky. Once set, only SET_PSV can clear it. 13 - block_guard_error: mismatch was detected in the CRC calculation for DIF or XOR,CSUM,CRC,for all other signature types. This bit is sticky. Once set, only SET_PSV can clear it.

Table 694 - Persistent Signature Variable (PSV) Fields

	Bits	Name	Description
	16	status	2:0 - next byte address alignment. If != 0, last transaction ended in unaligned address. 3 - app_tag_all_ones 4 - ref_tag_all_ones 5 - learn next signature [unimplemented] 10 - saw_reference_tag_error 11 - saw_application_tag_error 12 - saw_block_guard_error else - reserved
08h		transient signature	
0Ch	31:0	reference_tag	Counter incrementing every signature zone
Extended			
10h - 1Ch	128	Extended PSV	Contains debug information. Retrieved by the QUERY_PSV command. Includes the Key and VA of the first error detection point related to this PSV. This address is in terms of source domain (memory domain on send operations) and is the address of the last byte of the block in error.

Only *Base* part (DWORDs 1-3) are passed to the HW with SET_PSV WQE and returned by GET_PSV WQE (DWORD 0 is returned as zeroes). QUERY_PSV command is used to retrieve the entire structure specified in Table 690 for each PSV inquired.

17.4.2.2 Byte Stream Format (BSF)

Byte Stream Format (BSF) describes the input and output data structures in memory and wire domains. BSFs are configured through the UMR WQE. [Table 695](#) shows Byte Stream Format (BSF) layout.

Table 695 - Byte Stream Format (BSF) Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
bsf_size		same_block_structure						bsf_p_type		check_byte_mask						repeating block w_block_size_selector				repeating block m_block_size_selector /						0h	Basic				
																		w_pa d	w_check_gen w_init			m_pa d	m_check_gen m_init								
																								copy_byte_mask							
raw_data_size																												04h	Extended		
w_block_format_selector								w_block_psv_index																				08h			
m_block_format_selector								m_block_psv_index																				0Ch			
		w_t_check_gen w_t_init				m_t_check_gen m_t_init		prologue_size																				10h	Extended		
								d		epilogue_size																				14h	
w_transaction_format_selector										w_transaction_psv_index																				18h	
m_transaction_format_selector								m_transaction_psv_index																				1Ch			

Table 695 - Byte Stream Format (BSF) Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										
w_refresh_dif_params w_inline_vld																w_dif_app_tag											20h	inline			
		w_dif_ref_tag																									24h				
		w_block_format																									28-2Ch				
m_refresh_dif_params m_inline_vld																m_dif_app_tag											30h				
		m_dif_ref_tag																									34h				
		m_block_format																									38-3Ch				

m_ and **w_** prefix in field name indicates whether this parameter specifies memory or wire side of the BSF. Bits 0-15 of the first Dword of the BSF can have different meanings based on the type of block signature and the same_block_structure field.

Table 696 specifies fields of the Byte Stream Format.

Table 696 - Byte Stream Format (BSF) Fields

Bits	Name	Description
2	bsf_size	0x0: BSF_16B - <i>Extended</i> part of BSF does not exist. (BSF size is 16B) 0x1: BSF_32B- <i>Extended</i> part of BSF exists (BSF size is 32B) 0x2: BSF_64B- <i>Extended</i> and <i>inline</i> sections exist or Crypto only section exists (BSF size is 64B) 0x3: BSF_128B - Includes both Signature and Crypto BSF parts (each of 64B). Signature 64B must go first, followed by 64B of Crypto BSF.
1	same_block_structure	When set, indicates that wire and memory have the same structure (block and signature size). Qualifies copy_byte_mask
2	bsf_p_type	Type of this BSF chunk (64B): 0x0: Signature 0x1: Crypto

Table 696 - Byte Stream Format (BSF) Fields

Bits	Name	Description
8	check_byte_mask	Each bit corresponds to a byte of the signature field in input domain. Byte of the input signature is checked if corresponding bit in the check_byte_mask is set. Bit 7 corresponds to transient_signature bits 31:24 Bit 0 corresponds to reference_tag bits 7:0 Bits not relevant to the signature type are ignored.
8	copy_byte_mask	Each bit corresponds to a byte in the signature field. If bit is set, corresponding byte in signature should be copied to destination domain. Bit(s) can be set only if source and destination signatures' format and block sizes match. This is handy to implement "check-only" as well as forwarding meta-data (along with check_byte_mask field). Bit 7 corresponds to transient_signature bits 31:24 Bit 0 corresponds to reference_tag bits 7:0 Valid only when "same_block_structure" bit is set. Bits not relevant to the signature type are ignored.
8	_block_size_selector	Block size index in block sizes table. Valid for repeating block signatures only.
2	_pad	Specifies signature alignment padding for block signature. This field specifies number of bytes that should be stripped/added from/to the end of the data block before validating/generating the signature of the source/destination domain. These bytes are covered by the signature. Valid for variable block signatures only. Padding will only checked or generated when check_gen is set.
1	_init	Initialize Block signature with its default value upon start of processing this segment. Valid for variable block signatures only.
1	_check_gen	Block signature is expected and validated on source, or will be appended on destination after the last byte of this segment. Valid for variable block signatures only.
1	_t_init	Initialize Transaction signature with its default value when start processing this segment
1	_t_check_gen	Transaction signature is expected on input or will be appended on output after the last byte of this segment
32	raw_data_size	Number of bytes of Raw data covered by this BSF. Zero means zero bytes of raw data (this can be used to transfer "signature only" or extend DUMP/FILL operations beyond what is supported in the BSF format).
24	prologue_size.	Number of bytes of Raw data (in addition to that specified in raw_data_size) to be dumped or filled at the beginning of the data defined by this BSF. Valid only on memory side of the signature handover operation. Zero value means no data to be dumped or filled.

Table 696 - Byte Stream Format (BSF) Fields

Bits	Name	Description
24	epilogue_size	Number of bytes of <i>Raw</i> data (in addition to that specified in <i>raw_data_size</i>) to be dumped or filled at the end of the data defined by this BSF. Valid only on memory side of the signature handover operation. Zero value means no data to be dumped or filled.
8	_block_format_selector	Index in the signature rules table for the block signature type. Zero value in this field means no signature. 0xFF value of this field is RESERVED. The layout of Signature rules Table entry is shown in Table 699 .
8	_transaction_format_selector	Index in the signature rules table for the transaction signature type. Zero value in this field means no signature. 0xFF value of this field is RESERVED. See Table 701, “Signature Rules Table,” on page 910
24	_block_psv_index	PSV handle for block signature
24	_transaction_psv_index	PSV handle for transaction signature
64	_block_format	Inline Block. format. The layout of Signature block format is shown in Table 699 .
1	_inline_vld	If set, block_format field in the inline section is valid. Block format is taken from the inline block format in the BSF and not from the table.
1	_refresh_dif_params	Must be set in DIF, for each BSF that has no relation with previous BSFs. “Has its own app_tag and ref_tag, and not a continuance of previous BSF”.
16	_dif_app_tag	Valid only when refresh_dif_params =1, DIF application tag.
32	_dif_ref_tag	Valid only when refresh_dif_params =1, initial DIF reference tag.

Signature Context Table is 256-entry table that contains signature rules supported by the device. The format of Signature Table entry is shown in [Table 699](#). The signature rules table is shown in [Table 701](#).

Table 697 - Crypto Byte Stream Format (BSF) Layout

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0												00h
bsf_size						bsf_p_type						encryp- tion_order														encryp- tion_stan- dard							
raw_data_size																														04h			
crypto_block_- size_pointer																														08h			
																														0Ch			
xts_initial_tweak																														10h-1Ch			
							dek_pointer																							20h			
																														24h			
keytag																														28h-2Ch			
																														30h-3Ch			

Table 698 - Crypto Byte Stream Format (BSF) Fields

Offset	Bits	Name	Description
00h	31:30	bsf_size	0x0: BSF_16B - Extended part of BSF does not exist. (BSF size is 16B) 0x1: BSF_32B- Extended part of BSF exists (BSF size is 32B) 0x2: BSF_64B- Extended and inline sections exist or Crypto only section exist (BSF size is 64B) 0x3: BSF_128B - Includes both Signature and Crypto BSF parts (each of 64B). Signature 64B must go first, followed by 64B of Crypto BSF.
	25:24	bsf_p_type	Type of this BSF chunk (64B): 0x0: Signature 0x1: Crypto
	19:16	encryption_order	Crypto encrypt/decrypt and order respective to signature: 0x0: Encrypt_wire_signature 0x1: Encrypted_memory_signature 0x2: Encrypted_raw_wire - (no memory signature) 0x3: Encrypted_raw_memory - (no wire signature)
	3:0	encryption_standard	0x0: AES_XTS
04h	31:0	raw_data_size	Number of bytes of Raw data covered by this BSF.
08h	31:24	crypto_block_size_pointer	Pointer to a hardware table indicating block size
10h-1Ch	128	xts_initial_tweak	
20h	23:0	dek_pointer	Pointer to DEK, 24 LSBs of DEK object ID
28h-2Ch	64	keytag	Keytag of DEK (clear text)

Table 700 - Signature Rules Fields

Offset	Bits	Name	Description
00h	31:24	sig_type	encoded signature type: 0x0: unprotected 0x1: T10-DIF 0x2: T10-DIF CS (checksum) 0x3: CRC32 0x4: reserved 0x5: XOR32 0x6: XOR8 0x7: non-specific (unprotected, but PSV is valid for zero detect, init) 0x8: CRC64_xp10 - Supported when HCA_ - CAP.sig_crc64_xp10==1. 0x9: CRC32c - Supported when HCA_CAP.sig_crc32c==1. 0xA - 0xFF - reserved
23		block_sigs_covered / repeating_block	For transaction signatures: block_sigs_covered: Transaction signature covers the signature zones. For block signatures: repeating_block: Block signature is a repeating type
22		invert_crc invert_crc	invertinvert CRC before injecting/checking it
21		swap_crc_bits	swap the order of the CRC bits before check/transmit. this is done at the byte level [Internal] debug hook
20		swap_crc_bytes	swap the order of the CRC bytes before check/transmit [Internal] debug hook
19		seed	invert_crc==0: 0 - 0x0, 1 - 0xFFFF invert_crc==1: 1 - 0x0, 0 - 0xFFFF Number of '1' in the seed fits the signature width according to the signature type
18		invert_data_bits	Invert the data bits before calculating signature [Internal] debug hook
17		swap_data_bits	Swap the order of the data bits in every byte before calculating signature [Internal] debug hook

Table 700 - Signature Rules Fields

Offset	Bits	Name	Description
04h	22	dif_inc_ref_tag	For DIF, when set increment reference tag per block
	1	dif_learn_ref_tag	[unimplemented - HW not verified] when DIF learning enabled, copy reference tag from source for 1st block, use it for following blocks.
	1	dif_learn_app_tag	[unimplemented - HW not verified] when DIF learning enabled, copy application tag from source for 1st block, use it for following blocks.
	2	dif_learn_mode	[unimplemented - HW not verified] 0x0 - never learn 0x1 - learn first block signature in BSF 0x2 - learn next block signature in BSF on read response first for responder only 0x3 - learn first block signature OR next block signature in BSF on read response first for responder only
	17:16	dif_guard_check_type	0x0 - always check guard 0x1 - do not check guard if application tag is 0xffff 0x2 - do not check guard if application tag is 0xffff and reference tag is 0xffffffff 0x3 - reserved
	15:0	dif_app_tag_bit_check_en	Bit mask for checking the application tag for T10 DIF. Set bits would be validated against the PSV. For bit mask in byte granularity, check_byte_mask[5:4] could be used instead.

[Internal] In the following Signature Rules table the following fields are always 0x0 and are not exposed: swap_crc_bits, swap_crc_bytes, swap_data_bits, invert_data_bits. Also, the field invert_crc is renamed to corrupt_crc since the bit also causes the seed to be inverted as well.

Table 701 - Signature Rules Table

Index	Signature Type	Repeating / Signature Zones Covered	Seed	invert Invert crc	Increment Reference Tag	DIF Guard Check Type	DIF Application Tag Check Mask
0	un-protected	N/A	N/A	N/A0	N/A	N/A	N/A
1	non specific	N/A	N/A	N/A0	N/A	N/A	N/A
2-3	reserved	N/A					
4	T10 DIF	1	0	0	1	0x0	0xffff

Table 701 - Signature Rules Table

Index	Signature Type	Repeating / Signature Zones Covered	Seed	invert Invert crc	Increment Reference Tag	DIF Guard Check Type	DIF Application Tag Check Mask
5	T10 DIF	1	1	0	1	0x0	0xffff
6	T10 DIF	1	0	1	1	0x0	0xffff
7	T10 DIF	1	1	1	1	0x0	0xffff
8	T10 DIF	1	0	0	1	0x1	0xffff
9	T10 DIF	1	1	0	1	0x1	0xffff
10	T10 DIF	1	0	0	1	0x1	0x00ff
11	T10 DIF	1	0	0	1	0x1	0xff00
12	T10 DIF	1	0	0	0	0	0xffff
13	T10 DIF CS	1	0	0	1	0	0xffff
14	T10 DIF CS	1	0	0	0	0	0xffff
15	reserved	N/A					
16	XOR8	0	0	0	0	N/A	N/A
17	XOR8	0	1	0	0	N/A	N/A
18	XOR8	1	0	0	0	N/A	N/A
19	XOR8	1	1	0	0	N/A	N/A
20-31	reserved	N/A					
32	CRC32	0	0	0	0	N/A	N/A
33	CRC32	0	1	0	0	N/A	N/A
34	CRC32	1	0	0	0	N/A	N/A
35	CRC32	1	1	0	0	N/A	N/A
36-50	reserved	N/A					
51	T10 DIF	1	0	0	0	1	0xffff
52	T10 DIF CS	1	0	0	1	1	0xffff

Table 701 - Signature Rules Table

Index	Signature Type	Repeating / Signature Zones Covered	Seed	invert Invert crc	Increment Reference Tag	DIF Guard Check Type	DIF Application Tag Check Mask
53	T10 DIF CS	1	0	0	0	1	0xffff
54-63	reserved	N/A					
64	CRC32c	0	0	0	0	N/A	N/A
65	CRC32c	0	1	0	0	N/A	N/A
66	CRC32c	1	0	0	0	N/A	N/A
67	CRC32c	1	1	0	0	N/A	N/A
68-79	reserved	N/A					
80	CRC64 _xp10	0	0	0	0	N/A	N/A
81	CRC64 _xp10	0	1	0	0	N/A	N/A
82	CRC64 _xp10	1	0	0	0	N/A	N/A
83	CRC64 _xp10	1	1	0	0	N/A	N/A
84-255	reserved	N/A					

Block Size Table is 256-entry table that contains block sizes supported by the device. See [Table 702, “Block Size Table,” on page 912.](#)

Table 702 - Block Size Table

Index	Block Size in Bytes	Comment
0	reserved	
1	512	
2	520	
3	4096	

Table 702 - Block Size Table

Index	Block Size in Bytes	Comment
4	4160	
5	1M	
6	4048	Supported only when HCA_ - CAP.sig_ - block_4048== 1
6-255	reserved	

17.4.3 Signature Handover Work Requests

17.4.3.1 UMR Work Request

BSFs are configured via the UMR Work Request. See “User-Mode Memory Registration (UMR)” on page 918.

17.4.3.2 SET_PSV Work Request

PSVs are configured via the SET_PSV Work Request. See Table 58, “SET_PSV Segment Format,” on page 327.

17.4.3.3 CHECK_PSV Work Request

The consolidated status of multiple PSVs can be retrieved (read) from the device by posting CHECK_PSV (pronounced ‘CHECK_PSVzzz’) Work Request on Send Queue. CHECK_PSV returns consolidated status of signature operation results of a list of PSVs specified in the WQE. The data segment includes a pointer to the memory location to which the CHECK_PSV result will be written to. Note that the CHECK_PSV CQE does not contain an indication of the result of the PSVs checking.

[Internal - implemented by HW, not supported] CHECK_PSV can be configured to cause a transport error in the case that the check failed. This is done through the transport_err_mask and the transport_err_operation fields. transport_err_mask field is used to mask some of the bits of the coalesced status. AND or OR operations are applied to the other bits based on the transport_err_operation field in a similar manner to the error_coalescing_operation field.

CHECK_PSV Work Request is shown in Section 8.9.4.1.8, “CHECK_PSV/rCHECK_PSV Segment,” on page 328. CHECK_PSV response is scattered to the memory location pointed by the data seg-

ment in the CHECK_PSV WQE. The format of the response is shown in Table 703, “CHECK_PSV Response Layout”.

Table 703 - CHECK_PSV Response Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0		seg
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											response
																												00-38h				
																consolidated_status												3Ch				

Table 704 - CHECK_PSV Response Fields

Offset	Bits	Name	Description
00h-38h		reserved	
3Ch	31:16	reserved	
	15:0	consolidated_status	A consolidated status of several PSVs is created by a bit-wise operation on their syndrome field (see Table 694, “Persistent Signature Variable (PSV) Fields,” on page 900). The bit-wise operation is controlled by the error_coalescing_operation field. If bit error_coalescing_operation[i] is set to 1, bit ‘i’ across all, PSVs will be ‘OR’ed, otherwise, AND operation will be used. For example, if PSV0.syndrome = 0xCC, PSV1.syndrome = 0xAA and WQE.error_coalescing_operation = 0xF0, then consolidated_status would be 0xE8.

Execution of this command is subject to checking of the Protection Domain (PD). Each one of the PSVs specified in the WQE must have a matching PD to the one specified in the Send Queue. If any of them does not match, the WQE would be completed with error.

17.4.3.4 GET_PSV Work Request

Persistent Signature Variables can be retrieved (read) from the device by posting GET_PSV Work Request on Send Queue. GET_PSV returns up to 4 PSVs specified in the WQE.

Format of GET_PSV Work Request is shown schematically in Section 8.9.4.1.9, “GET_PSV/RGET_PSV Segment,” on page 330.

Format of output data is shown in Table 705. If a signature error occurred, the extended PSV information can be retrieved by the QUERY_PSV command. See Section 29.7.3, “QUERY_PSV – Query PSV,” on page 1879.

Table 705 - GET_PSV Work Request Output

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
PSV0 (Table 690, "Persistent Signature Variable (PSV) Layout")																															00-0Ch
PSV1 (Table 690, "Persistent Signature Variable (PSV) Layout")																															10-1Ch
PSV2 (Table 690, "Persistent Signature Variable (PSV) Layout")																															20h-2Ch
PSV3 (Table 690, "Persistent Signature Variable (PSV) Layout")																															30h-3Ch

Send QueueExecution of this command is subject to checking of the Protection Domain (PD). Each one of the PSVs specified in the WQE must have a matching PD to the one specified in the Send Queue. If any of them does not match, the WQE would be completed with error.

17.4.3.5 RGET_PSV and RCHECK_PSV WQEs

RGET_PSV and RCHECK_PSV have the same format and output as the GET_PSV and CHECK_PSV respectively. However, they enable special fence mode. With this new fence mode, the specific RGET_PSV or RCHECK_PSV WQEs will not be executed before the former WQE is completed, however, the following WQEs (after the RGET or RCHECK) do not require fencing to assure the execution order.

Figure 89: RGET_PSV Example

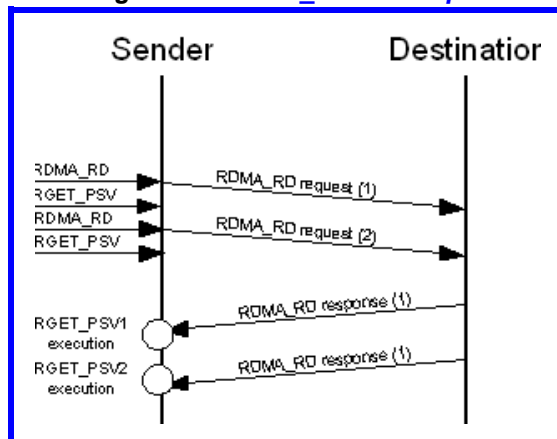


Figure 89, “RGET_PSV Example” shows an example of 4 WQEs posted on a Send Queue. In this example, RGET_PSV1 is fenced by RDMA_RD 1 and will be executed only when RDMA_RD1 is done. However, it does not fence RDMA_RD2 execution. This way, it enables having the two RDMA_RD operations to be executed simultaneously.

17.4.3.6 DUMP Work Request

DUMP Work Request can be used to execute signature handover operation (or validate signature of memory-resident data structure) without transmitting the data on the wire. This Work Request processes data just as if it were transmitted, creating/updating/checking signatures using the rules set by the BSF(s). When complete, the PSVs specified in the BSF(s) will contain the signature results. This Work Request can also be used to calculate initial value of the signature for subsequent data-transferring signature operation. This can be handy for debug and signature error recovery.

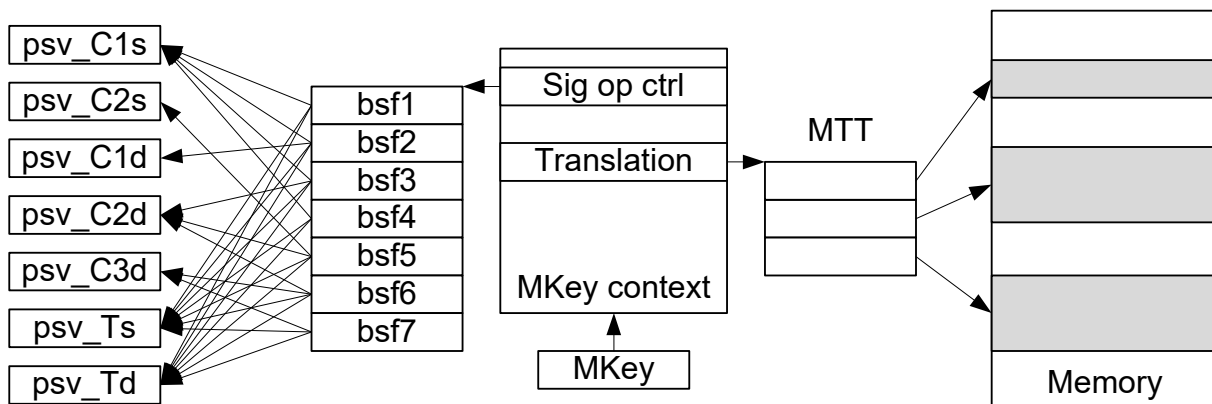
Format of DUMP Work Request is shown in [Section 8.9.4.1.10, “DUMP Work Request,” on page 331](#). Naturally, the Mkey specified in the data segment should have been previously coded for signature operations via a UMR WQE, otherwise this WQE would only accomplish generating a small amount of heat.

18 Address Translation and Protection Enhancements

18.1 Byte Stream Format

An MKey context can specify signature hand over parameters, in which case, in addition to the MTT or KLM entries, the MKey context specifies Byte Stream Formats (BSFs) which are used to control address calculation during a signature hand over operation. This is illustrated in [Figure 90](#).

Figure 90: Byte Stream Format MKey



For details about signature hand over operation please refer to [“Application Data Integrity: Signature Handover Operations”](#) on page 880.

18.2 Dump and Fill Memory Key

HCA deploys special memory keys for special memory operations.

Dump and fill memory key can be used as a destination for memory writes that will not update memory. Write memory access that uses the “dump and fill” memory key will not show up on the PCI bus. Read memory access that uses the “dump and fill” memory key will generate all zeroes for the entire request.

DUMP & FILL memory key is obtained using QUERY_SPECIAL_CONTEXTS command.

18.3 Lightweight Memory Registration

Lightweight memory registration enables the creation of virtually-contiguous address spaces out of disjoint (non-contiguous) chunks of memory region(s) already registered with the HCA.

With the adapter device, this is done with a User Mode Memory Registration (UMR) Work Request posted to a SQ. This is the “right way” to execute lightweight memory registration on the device. UMR is a non-privileged operation enabling creation of a virtually contiguous memory region on byte granularity. This is the fastest and recommended way to execute light memory registration on the device.

UMR Work Request is also used for invalidation of memory region/window.

UMR is described below in [“User-Mode Memory Registration \(UMR\)”](#) on page 918.

18.3.1 User-Mode Memory Registration (UMR)

UMR is a mechanism to alter the address translation properties of MKeys by posting Work Requests on SQs. The key advantage of this mechanism versus prior fast registration implementations is that this operation can be executed by non-privileged software and the granularity (size and alignment) of memory sections specified by the KLM entries are not constrained (in contrast to page or block granularity previously required).

UMR can also alter some of the original MKey fields. MKey fields to update are marked in the MKeyCtx mask bit (in the UMR control segment) and their new values are taken from the MKeyCtx section in the UMR WQE.

Not all MKey fields can be changed by a UMR request. HW will block changing fields that should not be changed. For the list of fields that can be changed, see [Table 53, “UMR Control Segment Fields,” on page 323](#).

UMR can be executed on a memory region only if the ***umr_en*** bit in Mkey Context is set.

An indirect memory key, when initially created, has no virtual to physical mapping and cannot be used until it is mapped to physical addresses by executing a UMR request on one of the SQs.

UMR uses the MKey index as an argument and fills in its corresponding memory translation table entries from a list of virtual buffers specified in the UMR request. [TBD MOVE THIS TO WQE chapter](#)

UMR specifies access rights for the Mkey being configured. If HCA_CAP.imaicl=0, in the course of address translation, HW validates that the access rights in an indirect MKey do not exceed the access rights of the MKey pointed by it. An access right (e.g, remote write) is granted if, and only if, it is allowed by all MKeys processed in the course of this address translation. Thus, bogus programming of access rights to an indirect MKey is discovered during a memory access that uses this key and will be reported as a memory access error and not as UMR programming error.

If HCA_CAP.imaicl=1 however, an MKey accessed through an indirect MKey need not have remote access rights. This allows, for instance, to implement Memory Window (implemented by an indirect mkey) with remote access bound to a Memory Region (implemented by a direct mkey) with no remote access rights. Kernel protocol drivers can also use this ability to register indirect MKeys with remote access on top of the reserved LKey.

The new translation given by a UMR request takes effect prior to the execution of the next command on the same WQ (this WQE should be marked with small-fence), thereby enabling the posting of a UMR followed by a WQE using the same key in a single operation.

UMR requests of a single indirect MKey can be performed repeatedly. Each UMR request remaps the MKey's translation entries. Unlike previous implementations, software does not need to handle the synchronization of the HCA's cache. Changing the variant field of the MKey can be done by software for security purposes, but no cache flush or other synchronization operations are required for the repeated use of same MKey.

18.3.1.1 Strided Block Format

UMR argument can contain, in addition to the buffers description list (aka scatter/gather entries or KLM), control blocks that extend memory management flexibility beyond banal address translation. The format of the control block is shown in [Table 706](#).

The repeat operation supported in the device effectively allows the programmer to interleave elements of multiple arrays into a single cohesive array with a single command.

Table 706 - Repeated Block Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00
bytecount_per_cycle																												04				
repeated_block_mkey																												08				
repeat_count																												0C				
reserved														num_entries														10h-1Ch				
repeated_block_entry[0]																												20h-2Ch				
repeated_block_entry[1]																												30h-...				
...																																

Table 707 - Repeated Block Fields Description

Offset	Bits	Name	Description
00h	31:0	byte_count_per_cycle	Specifies the accumulated byte count of all entries.
04h	31:0	repeated_block_mkey	Constant value that indicates a repeated block format. This value is reported in repeated_block_mkey in QUERY_SPECIAL_CONTEXT See Table 1819, “QUERY_SPECIAL_CONTEXTS Input Structure Layout,” on page 1871
08h	31:0	repeat_count	Specifies number of times operation is to be repeated.
0Ch	15:0	num_entries	Specifies number of UMR blocks affected by this control block. Those entries includes both <i>bytecount</i> and <i>stride</i> . The block of size <i>bytecount</i> resides in the beginning of a <i>stride</i> . The next block will be in the following stride.
10h-...	128	repeated_block_entry[...]	List of repeated block entries. See Table 709, “Repeated Block entry Fields Description,” on page 920

Table 708 - Repeated Block Entry Format

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
stride																byte_count																00
mem_key																																04
address																																08h-0Ch

Table 709 - Repeated Block entry Fields Description

Offset	Bits	Name	Description
00h	31:16	stride	The address gap between the first byte of this block in 2 consecutive cycles
	15:0	byte_count	The number of bytes in a single block. May not be greater than stride
04h	31:0	mem_key	The Memory key mapping this block in all cycles
08h-0Ch	64	address	The address of the first byte of the block in the first cycle

Taking Data from Alternating Arrays Example

For instance, 100 alternate access to 100 members of two arrays (one of 512 bytes and another one 8 bytes) can be programmed in three UMR blocks as shown below:

```
ctrl.repeat_count = 100;           // repeat 100 times
ctrl.num_entries = 2;              // 2 entries in a cycle
ctrl.bytestream_per_cycle = 520;
entry0.ByteCount = 512;            // Individual element in the buffer is 512 bytes
entry0.stride = 512;               // stride equal to block size
entry0.MKey = MKey1;              // this is data buffer segment describing the first array
entry0.Address = Adr1;             // start address of first array
entry1.ByteCount = 8;              // Individual element in the buffer is 8 bytes
entry1.stride = 8;                 // stride equal to block size
entry1.MKey = MKey2;              // this is data buffer segment describing the second array
entry1.Address = Adr2;             // start address of second array
```

Programming UMR with argument above will result in alternating access of one element of first array and then one element of second array.

Stride Data Example

In the following example a single column in a two-dimensional matrix is mapped. The matrix size is 16x32, each entry is 8B, and the matrix is arranged in memory by rows (e.g. first row is in addresses 0 - 255, second row starts at 256). The chosen column is the first column. This data type is built by a list of blocks (matrix entry), separated by a fixed stride (matrix row).

```
ctrl.repeat_count = 16; // number of rows in the matrix
ctrl.num_entries = 1; // 1 memory location per cycle
ctrl.bytecount_per_cycle = 8; // 1 entry with ByteCount 8
entry0.ByteCount = 8; // Individual entry in the matrix
entry0.stride = 256; // distance between 2 consecutive entries in a column (or row size: 8B x 32)
entry0.MKey = MKey1; // MKey of the matrix
entry0.Address = Adr1; // start address of entry Matrix[0,0]
```

18.3.1.2 UMR Common Programming Model

The following is a common programming procedure for a UMR:

- Create memory key with *indirect* or *direct* property, allocating enough translation space for this key (see [“Memory Key Configuration \(Creation\)” on page 634](#)).
- In the critical data path:
 - Post a UMR Work Request on the SQ that is destined to use the new mapping. (MKey should be at a valid and free state).
 - Post a WQE that uses the newly mapped MKey through local or remote access
 - Optionally, upon completion of the operation, invalidate the MKey in the HCA by posting an additional UMR which sets the free bit, or using send with invalidate command from remote node

18.3.1.3 UMR Global Pre-Registration Model

The device supports a global UMR pre-registration programming model. With this model, the driver can pre-allocate a global, non-PD constrained, pool of Mkeys. These Mkeys are created without translations and should have both the “umr_enable” and “free” bits set. Each such Mkey is created with a fixed size which determines the maximum number of translations it can contain. For greater flexibility, several such pools can be created with various sizes.

When a registration request is received from the user, the above pool of Mkeys can be used. This is accomplished by posting a UMR which: populates Mkey with translations, modifies PD to appropriate value and clears the free bit. Registration is considered complete once a completion has been received for the UMR WQE.

The QP/SQ on which these UMRs are posted should be with the REG_UMR transport type. Only a single QP/SQ should be configured to have this type and only UMR WQEs can be posted on it.

Upon completion of the operation, the user must invalidate the Mkey by posting a UMR setting the free bit.

18.3.1.4 Bind Memory Window Implementation with UMR

Memory Windows are created (allocated) via the command interface. When allocating a Memory Window, the software should issue a CREATE_MKEY command to CREATE MKey to be used as a window key. The only values that need to be specified in the command are:

- *access_mode=0x2*
- *for MW type 2:*
- *free=1*
- *en_rinval=1*
- *for MW type 1:*
- *free=0*
- *en_rinval=0*
- *translations_octword_size=1*
- *pd*
- *QPN=0xffffffff - for Type1 MW, QP number for Type2 MW*
- *umr_en* should be 1

Binding of memory region will be done by posting UMR Work Request (with single sge in the data segment) using the following parameters:

- *Control_Segment.UMR_mkey* = Window memory key
- *UMR_Segment.UMR_Control_Segment.check_free* = 1 (for Type2B), 0 for Type1.
- *UMR_Segment.data segment.sge.MKey* = Region memory key
- *UMR_Segment.data segment.sge.VA* = window start virtual address
- *UMR_Segment.data segment.sge.size* = window size

The MKey context and its bit mask (in the UMR control segment) must be set properly as well. During address translation on a memory window, the PD of the QP/WQ is checked against the PD of the window and the PD of the region. Consistency of the PD of the region to the PD of the window is not checked at bind time.

If a memory region is invalidated when there are memory windows still bounded to it, those windows will be left as “orphaned windows”. Address translation of memory access to them will fail. According to the InfiniBand Architecture Specification, it is allowed to create a memory window with remote access rights on top of a memory region with only local access rights. Thus when QUERY_HCA_CAP.imaicl=0 memory windows cannot be fully supported.

18.3.1.5 Local Invalidate Implementation with UMR

A local invalidate operation allows privileged and non-privileged applications to post a Work Request that alters the protection table. After successful execution, the region that was pointed to by this Work Request is invalidated and its state is changed to FREE.

The only field that should be specified is:

- UMR.free = 0

Local invalidation is guaranteed to take effect prior to a CQE for the operation it created.

18.3.1.6 UMR Work Request Operation

UMR Work Request modifies a variant part of the MKey and changes its MTT entries, loading a new list of buffers that will be used for address translation. For invalidation by UMR, however, no new buffer translation entries are loaded.

Hardware executes the following steps when executing a UMR Work Request:

1. Checks whether the MKey is valid
2. Checks whether the MKey is *free* (*N.A. to local-invalidation by UMR*)
3. Checks whether the PD of the SQ matches the PD of the MKey
4. If Mkey.QPN!=0xFFFFFFFF (Type 2 windows), checks whether the MKey.QP is the QP that is doing the UMR
5. Checks that the MKC.umr_en=1
6. Checks that buffers' list specified by the WQE does not exceed the memory allocated for that MKey
7. If all checks are successful, buffers' list specified by WQE is copied to MTT and xt fields are updated
8. Reports completion on the WQE. If some of the checks failed, completion with error is reported along with advised action to be applied on the programmer.

The protection checks are executed at every stage, either *when an indirect* MKey or a *direct* MKey is used.

18.4 MEMIC - Memory Mapped to InterConnect

The device supports using its internal buffers for SW defined data. The memory is referenced through an address in the HCA BAR or by Memory Keys (Mkeys) and therefore accessible through the device, the network or the PCIe. This memory is referred to as Memory Mapped to InterConnect (MEMIC).

The support for MEMIC is indicated with HCA_CAP.Device_Memory.*memic* (See Table 1605, "Device Memory Capabilities Layout," on page 1785).

Prior to using the MEMIC buffers, they should be allocated as explained in Section 18.4.1, "MEMIC Allocation," on page 924. Once allocated the buffer can be used to host data for packet transmission, control objects and so on.

The allocated *memic* can be registered by Mkey as explained in Section 18.4.3, "Memory Registration," on page 924.

Note: When accessing *memic*, all ordering requirements should be considered according to the interfaces used for buffer access and the appropriate mechanisms should be applied.

18.4.1 MEMIC Allocation

A MEMIC buffer is allocated using `ALLOC_MEMIC` command (see Section 29.6.6, “`ALLOC_MEMIC` – Allocate MEMIC,” on page 1873). The MEMIC can be allocated in the range `memic_bar_start_addr-- memic_bar_start_addr+2^log_memic_bar_size` in Table 1605, “Device Memory Capabilities Layout,” on page 1785, and the allocated buffer is contiguous.

SW should set the range where the required *memic* should be allocated - the start of range id indicated by *range_start_addr*, the size of the range is indicated *range_size*, and the size of the required buffer is indicated by *size* field. Both the *range_start_addr*, *range_size* and the required memic size must be aligned to 64 Bytes.

Note that the allocation of *memic* can fail and return `STATUS=NO_RESOURCES`. SW can try to reallocate with other allocation parameters (range and size).

18.4.2 MEMIC DE-Allocation

An allocated MEMIC buffer is freed using the `DEALLOC_MEMIC` command (see Section 29.6.6, “`ALLOC_MEMIC` – Allocate MEMIC,” on page 1873). The size of the freed buffer should be indicated by *size* field. The start address of the buffer should be indicated by *start_address* field.

All relevant MKEYs and control objects must be destroyed prior to the release of the MEMIC buffer. All MEMIC buffers must be deallocated prior to driver teardown.

18.4.3 Memory Registration

Once a memory buffer is allocated, it can be registered using `CREATE_MKEY` command (See Section 29.6.1, “`CREATE_MKEY` – Create MKey Entry,” on page 1861). The MEMIC allocation should be indicated by `access_mode=memic`. The resisted memory buffer is indicated by *start_addr* and *size*.

When registering a Memory Key for MEMIC the following properties must be set:

- *umr_en* must be 0x0

bsf_en must be 0x0

19 Quality of Service

19.1 Transmit Scheduling

The device offers a flexible and hierarchical setting of packet transmission scheduling policies, which are enforced at runtime, enabling fine granularity of the Quality Of Service (QoS) control and delivery across multiple consumers.

The transmit scheduler is responsible for selecting the next Work Queue to service. Scheduling may be arranged in a hierarchical structure, allowing BandWidth (BW) allocation and rate limitation at different layers independently.

Transmit scheduling is done in an hierarchical fashion, where each hierarchy manages its transmit resources independently. The different layers are implicitly merged into a single scheduling scheme by the device.

19.1.1 Embedded Switch Scheduling

The embedded switch manages the traffic transmitted by the PF and the VFs. QoS between the functions is provided using the Transmit Scheduler mechanism, see [Section 8.26, “Transmit Scheduler,” on page 535](#).

The Transmit Scheduler allows building the following models:

- QoS between VPorts
- QoS between VPorts per Traffic Class

19.1.1.1 Para-VPort

For QoS purposes, the E-Switch manager may associate groups of SQs with VPorts. Such VPorts are treated as separate VPorts by the Transmit Scheduler. These VPorts are referred to as Para-VPorts.

An example for using a Para-VPort, is when the E-Switch manager provides networking services for a VM that also has a dedicated network interface through a VF. In that case, for QoS purposes, the traffic generated by the E-Switch manager on behalf of the VM should adhere to the same QoS attributes assigned to the VF.

The number of supported Para-VPorts is given by *max_qos_para_vport* field in QUERY_HCA_CAP.

A Para-VPort is created by “CREATE_QOS_PARA_VPORT - Create an E-Switch Scheduler Para-VPort” command, and destroyed by “DESTROY_QOS_PARA_VPORT - Destroy an E-Switch Scheduler Para-VPort”.

An SQ is associated with Para-VPort by CREATE_SQ or MODIFY_SQ commands. The Para-VPort identifier should be placed in the *qos_para_vport_number* field. SQ that is not associated with a Para-VPort, is associated with the VPort where the SQ was created.

When creating a scheduling element for the Para-VPort, the *element_type* should indicate Para-VPort.

Note: Para-Vports are only supported for TC 0. In addition SQs associated with a Para-Vport may only transmit packets mapped to priority 0x0 (which must be mapped to TC 0).

•

19.1.1.2 Configuration Examples

The following diagrams presents an examples of Scheduling schemes. Figure 91 shows arbitration between 3 VFs and the PF. Each VF is associated with a single green VPort, while the PF is associated with the blue VPort.

Figure 91: E-Switch Scheduler per Function
VPort 0 VPort 1 VPort 2 Vport 3

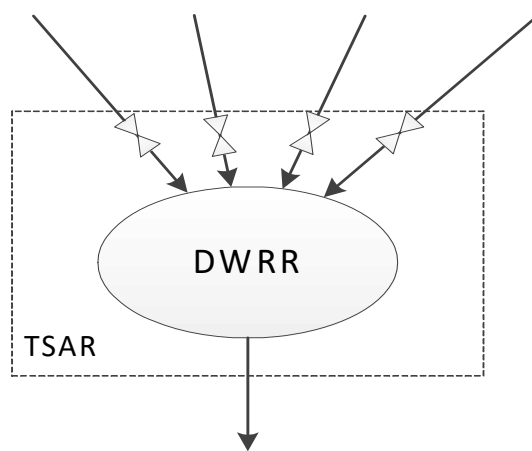
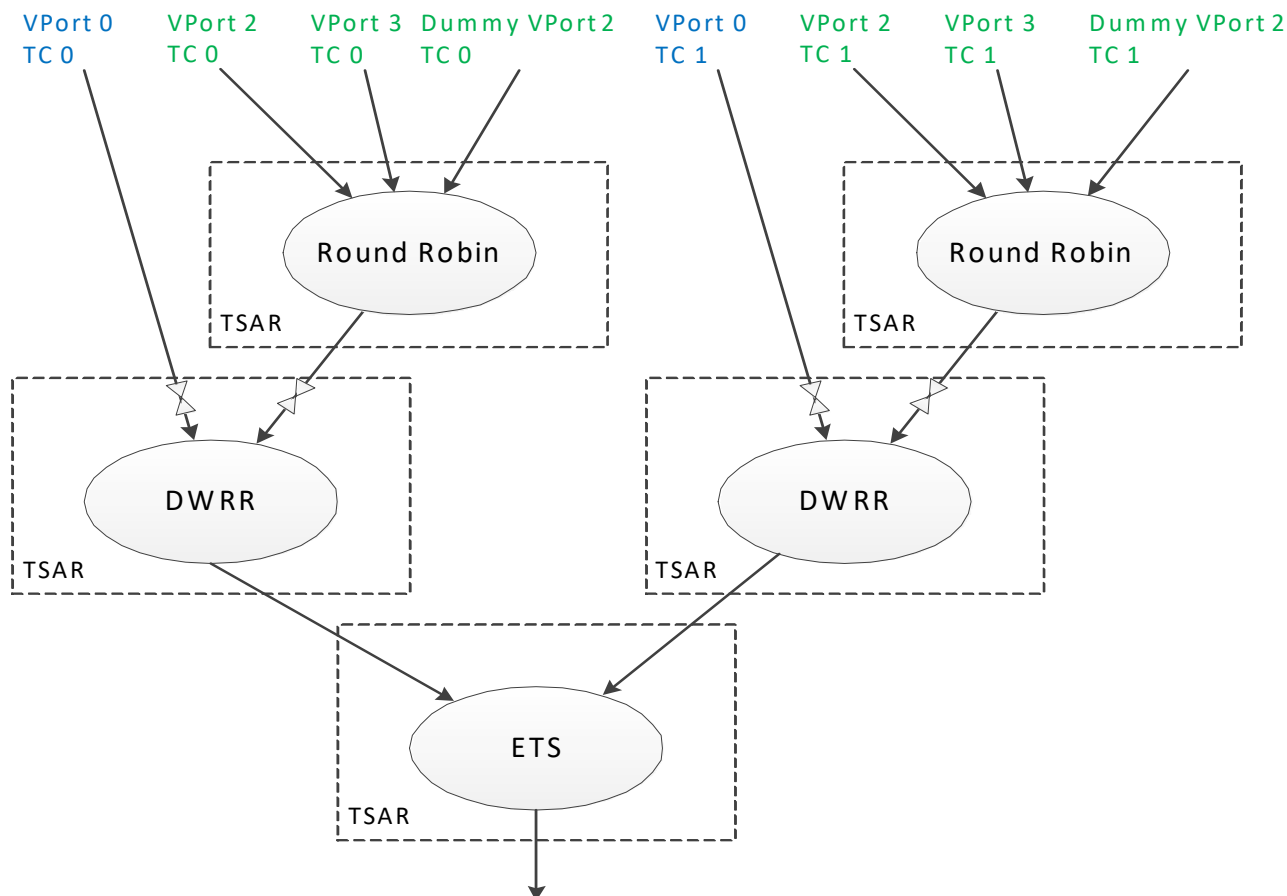


Figure 92 Shows arbitration between a VM and the Hypervisor. The VM is associated with the green VPorts and a Para-VPort which physically resides in the Hypervisor. The Hypervisor is associated with the blue VPort. Since the QoS attributes are defined per VM-TC:

1. The transmitting entities are separated per TC.
2. All transmitting entities belonging to the VM (for each TC) are first “aggregated” using a Round Robin Arbiter
3. Once QoS policy is applied per TC, arbitration between TCs is done using an ETS arbiter.

Figure 92: E-Switch Scheduler per VM-TC


19.1.1.3 Configuration Notes

The support for E-Switch scheduling is indicated by `HCA_CAP.esw_scheduling`. The supported scheduling elements are indicated by `HCA_CAP.esw_element_type` field. The supported TSAR types are indicated by `esw_tsar_type` field in the same command.

All scheduling elements created for E-Switch scheduling should be marked with `element_hierarchy` E_SWITCH.

The following scheduling elements are considered as scheduler “leaves”, and they must be the leaves of the scheduler tree:

- VPorts
- VPort-TC pair
- Para-VPort-TC pair

Scheduling elements of type VPort cannot be used in conjunction with the other types of leaf elements.

The number of TSARs should not exceed the total number of transmitting entities.

The E-Switch Transmit Scheduler supports a number of scheduler depths. Any path from the root TSAR to any of the leaf elements must be one of the following:

- A single DWRR TSAR.
- ETS TSAR (root) followed by a DWRR TSAR.
- ETS TSAR (root) followed by a DWRR TSAR followed by a Round Robin TSAR.
- A sequence of DWRR ~~or Round Robin~~ TSARs, so the path (excluding the leaf) is shorter or equal to $2^{\text{HCA\_CAP.log\_esw\_max\_sched\_depth}}$ but holds at least one TSAR.
- ETS TSAR (root) followed by a sequence of DWRR ~~or Round Robin~~ TSARs, so the path (excluding the leaf) is shorter or equal to $2^{\text{HCA\_CAP.log\_esw\_max\_sched\_depth}}$ but holds at least one TSAR.

19.2 Packet Pacing

This feature is supported when `QoS_CAP.packet_pacing==1`. See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773](#).

Send Queues (SQs) may be individually rate limited, thus, allowing granular rate control over specific SW-defined flows. A rate-limited flow is allowed to transmit a few packets (burst) before its transmission rate is evaluated, and the next packet is scheduled for transmission accordingly.

The rate limits should be configured to a table using `SET_PP_RATE_LIMIT` command. The size of the table is defined by `QoS_CAP.packet_pacing_rate_table_size`. The rate limit in index 0x0 is reserved for SQs which are not subject to packet pacing. The configured rate limits must not exceed the range defined by `QoS_CAP.packet_pacing_max_rate/packet_pacing_min_rate`.

When the rate limit is intended for a stream of similar packets, stating the typical packet size in `typical_packet_size` can improve the accuracy of the rate limiter. The expected packet size should be the same for all SQs associated with the same rate limit index. Support for stating the typical packet size is indicated by `HCA_CAP.packet_pacing_typical_size`.

When the size of the burst is critical, and must be bounded, an upper limit for the burst size may be set per rate limit in **`burst_upper_bound`** field. In other cases, the burst size will be calculated by the device defaults. It is defined to achieve best performance and accuracy for the rate limit. This feature is supported if `HCA_CAP.QoS.packet_pacing_burst_bound==1` is set (See [Table 1597, “Quality of Service Capabilities Layout,” on page 1773](#)).

The rate limit table may be queried using the `QUERY_PP_RATE_LIMIT` command. By default, the table is empty and its entries are invalid, except for the reserved entry.

The rate limit assigned to an SQ/QP is indicated by `packet_pacing_rate_limit_index` field in SQ context/**QP context**.

An SQ is associated with a rate limit at creation time using `CREATE_SQ` command, and may be modified at any stage by `MODIFY_SQ` command.

A QP is associated with a rate limit at creation time using `CREATE_QP` command, and may be modified at any stage by `MODIFY_SQ` command.

SQs/QPs must not be assigned with an invalid entry in the rate limit table.

When multiple SQs/QPs are associated with the same rate limit in the table, each SQ is individually limited to the specified rate.

Modifying a rate limit while SQs/QPs are assigned to it is not allowed.

19.3 QoS Control Registers

The following section lists the various registers used to configure/query the QoS parameters of the switch. [Table 710](#) summarizes switch register set for QoS control.

Table 710 - QoS Register Summary

Register Name	Register ID (hex)	Query	Write	HCA	InfiniBand	Ethernet	Reference
QCAP - QoS Capabilities Register	0x4001	v	-	-	-	-	Table 711, “QCAP - QoS Capabilities Register Layout,” on page 933
QPTS - QoS Priority Trust State Register	0x4002	v	v	v	-	v	Table 713, “QPTS - QoS Priority Trust State Register Layout,” on page 934
QDPM - QoS DSCP to Priority Mapping Register	0x4003	v	v	-	-	-	Table 715, “QDPM - QoS DSCP to Priority Mapping Register Layout,” on page 935
QPDPM - QoS Port DSCP to Priority Mapping Register	0x4013	v	v	v	-	v	Table 719, “QPDPM - QoS Port DSCP to Priority Mapping Register Layout,” on page 937
QEPM - QoS EXP to Priority Mapping Register	0x4014	v	v	-	-	-	Table 723, “QEPM - QoS EXP to Priority Mapping Register Layout,” on page 939
QPCR - QoS Policer Configuration Register	0x4004	v	v	-	-	-	Table 727, “QPCR - QoS Policer Configuration Register Layout,” on page 941
QPBR - QoS Policer Binding Register	0x400C	v	v	-	-	-	Table 729, “QPBR - QoS Policer Binding Register Layout,” on page 947
QETCR - QoS ETS TC Configuration Register	0x4005	v	v	v	-	v	Table 731, “QETCR - QoS ETS TC Configuration Register Layout,” on page 949
QEGCS - QoS ETS General Configuration Register	0x4006	v	v	-	-	-	Table 737, “QEGCS - QoS ETS General Configuration Register Layout,” on page 952
QEEC - QoS ETS Element Configuration	0x400D	v	v	-	-	-	Table 739, “QEEC - QoS ETS Element Configuration Register Layout,” on page 954
QPDP - QoS Port Default Priority	0x4007	v	v	v	-	v	Table 741, “QPDP - QoS Port Default Priority Register Layout,” on page 959

Table 710 - QoS Register Summary

Register Name	Register ID (hex)	Query	Write	HCA	InfiniBand	Ethernet	Reference
QPDPC - QoS Port Default PCP	0x4017	v	v	-	-	-	Table 743, “QPDPC - QoS Port Default PCP Register Layout,” on page 960
QPRT - QoS Priority Regeneration Table	0x4008	v	v	v	-	v	Table 745, “QPRT - QoS Priority Regeneration Table Register Layout,” on page 961
QSPTC - QoS Stacking Port TClass	0x4009	v	v	-	-	-	Table 747, “QSPTC - QoS Stacking Port TClass Register Layout,” on page 962
QTCT - QoS Switch Traffic Class Table	0x400A	v	v	v	-	v	Table 749, “QTCT - QoS Switch Traffic Class Table Register Layout,” on page 963
QTCTM - QoS Switch Traffic Class Table is Multicast aware	0x401A	v	v	-	-	-	Table 751, “QTCTM - QoS Switch Traffic Class Table is Multicast aware Register Layout,” on page 965
QSTCT - Switch Stacking TClass Table	0x400B	v	v	-	-	-	Table 753, “QSTCT - QoS Switch Stacking TClass Table Register Layout,” on page 966
QSPIP - Switch Priority to IEEE Priority Mapping Table	0x400E	v	v	v	-	v	Table 755, “QSPIP - QoS Switch Priority to IEEE Priority Register Layout,” on page 967
QSPCP - Switch Priority to Counting Priority Mapping Table	0x401E	v	v	v	-	v	Table 757, “QSPCP - QoS Switch Priority to Counting Priority Register Layout,” on page 968
QRWE - QoS ReWrite Enable	0x400F	v	v	-	-	-	Table 759, “QRWE - QoS ReWrite Enable Layout,” on page 969
QPEM - QoS Priority to EXP Mapping	0x4010	v	v	-	-	-	Table 761, “QPEM - QoS Priority to EXP Mapping Register Layout,” on page 969
QPDSM - QoS Priority to DSCP Mapping	0x4011	v	v	-	-	-	Table 769, “QPDSM - QoS Priority to DSCP Mapping Register Layout,” on page 971
QPPM - QoS Priority to PCP Mapping	0x4012	v	v	-	-	-	Table 775, “QPPM - QoS Priority to PCP Mapping Register Layout,” on page 973
QSLM - QoS Switch Lifetime Limit	0x4015	v	v	-	-	-	Table 781, “QSLM - QoS Switch Lifetime Limit Register Layout,” on page 974

Table 710 - QoS Register Summary

Register Name	Register ID (hex)	Query	Write	HCA	InfiniBand	Ethernet	Reference
QHLL - QoS Head of queue Lifetime Limit	0x4016	v	v	v	v	v	Table 783, “QHLL - QoS Head of queue Lifetime Limit Register Layout,” on page 976
QCAM	0x4019	v	-	v	v	v	Table 785, “QCAM - QoS Capabilities Mask Register Layout,” on page 977
QPSC	0x401B	v	v	-			Table 787, “QPSC - QoS PTP Shaper Configuration Register Layout,” on page 978 [Internal]
Reserved for NICs	0x4020-0x402F						[Internal]

19.3.1 QCAP - QoS Capabilities Register

The QCAP register reports the QoS capabilities of the device. [For Condor need to add link to Appendix G](#)

<TBD> This register is very preliminary and might change a bit until settled.

Table 711 - QCAP - QoS Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																								max_policers_per_port						08h		
																								max_policers_global						0Ch		
																								max_policers_cpu						10h		

Table 712 - QCAP - QoS Capabilities Register Fields

Offset	Bits	Name	Description	Access
08h	7:0	max_policers_per_port	Maximum number of policers available per port	RO
0Ch	7:0	max_policers_global	Maximum number of global policers	RO
10h	7:0	max_policers_cpu	Maximum number of cpu policers	RO

19.3.2 QPTS - QoS Priority Trust State Register

This register controls the port policy to calculate the switch priority and packet color based on incoming packet fields.

Table 713 - QPTS - QoS Priority Trust State Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
																															trust_stat e	04h	

Table 714 - QPTS - QoS Priority Trust State Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local Port	Index
04h	2:0	trust_state	0: TRUST_PORT - Reserved for NIC devices 1: TRUST_PCP - default 2: TRUST_DSCP 3: TRUST_BOTH - For NIC devices supported only when QCAM.QPTS_TRUST_BOTH == 1	RW

19.3.3 QDPM - QoS DSCP to Priority Mapping Register

This register is a global control for mapping from DSCP field to Switch Priority for IP packets. Configures all local ports. Configured also for data packets from CPU port.

A per port control is done by QPDPM, see [Section 19.3.4, “QPDPM - QoS Port DSCP to Priority Mapping Register,” on page 937](#). For switches supporting QPDPM the configuration of a port may change by the last write command of either QDPM or QPDPM.

When reading this register, the reading will be from the values of local_port ‘1’.

Table 715 - QDPM - QoS DSCP to Priority Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dscp[0]								dscp[1]								dscp[2]								dscp[3]								00h
dscp[4]								dscp[5]								dscp[6]								dscp[7]								04h
...																																08h-38h
dscp[60]								dscp[61]								dscp[62]								dscp[63]								3Ch

Table 716 - QDPM - QoS DSCP to Priority Mapping Register Fields

Offset	Bits	Name	Description	Access
00h-3Ch	31:24	dscp[64]	DSCP entry is the DSCP to Switch Priority mapping. For more details refer to Table 717 .	

Table 717 - dscp<i>i</i> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								e	color						switch_prio	00h

Table 718 - dscp<i>i</i> Fields

Offset	Bits	Name	Description	Access
00h	7	e	Enable update of the specific entry. When cleared, the <i>switch_prio</i> and <i>color</i> fields are ignored and the previous <i>switch_prio</i> and <i>color</i> values are preserved.	WO
	6:5	color	The new packet color value for the relevant DSCP value 0: GREEN 1: YELLOW 2: RED when	RW
	3:0	switch_prio	The new Switch Priority value for the relevant DSCP value	RW

19.3.4 QPDP - QoS Port DSCP to Priority Mapping Register

This register controls the mapping from DSCP field to Switch Priority for IP packets.

A global control is configured using QDPM, [Section 19.3.3, “QDPM - QoS DSCP to Priority Mapping Register,” on page 935](#). The configuration of a port may change by the last write command of either QDPM or QPDP.

Table 719 - QPDP - QoS Port DSCP to Priority Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch-7Ch
																																80h

Table 720 - QPDP - QoS Port DSCP to Priority Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local Port. Supported for data packets from CPU port.	Index
04h-80h	31:16	dscp[64]	DSCP entry is the DSCP to Switch Priority mapping. For more details refer to Table 721, “dscp<i> Layout,” on page 937	

Table 721 - dscp<i> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 722 - dscp<i>i</i> Fields

Offset	Bits	Name	Description	Access
00h	15	e	Enable update of the specific entry. When cleared, the <i>switch_prio</i> and <i>color</i> fields are ignored and the previous <i>switch_prio</i> and <i>color</i> values are preserved.	WO
	9:8	color	The new packet color value for the relevant DSCP value 0: GREEN 1: YELLOW 2: RED whenFor Switch only, Reserved for NICs	RW
	3:0	switch_prio	The new Switch Priority value for the relevant DSCP value For NIC devices only 8 priorities are supported	RW

19.3.5 QEPM - QoS EXP to Priority Mapping Register

This register controls the mapping from MPLS EXP field to Switch Priority, Packet Color and ECN for MPLS packets.

Table 723 - QEPM - QoS EXP to Priority Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
exp[0]																exp[1]																04h	
exp[2]																exp[3]																08h	
exp[4]																exp[5]																0Ch	
exp[6]																exp[7]																10h	

Table 724 - QEPM - QoS EXP to Priority Mapping Register Fields

Offset	Bits	Name	Description	Access
0h	23:16	local_port	Local port. Supported for data packets from CPU port.	Index
04h-10h	31:16	exp[8]	EXP entry is the EXP to Switch Priority mapping. For more details refer to Table 725, “exp<i> Layout,” on page 939	

Table 725 - exp<i> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																e					ecn	color									switch_prio	00h

Table 726 - exp<i> Fields

Offset	Bits	Name	Description	Access
00h	15	e	Enable update of the specific entry. When cleared, the <i>switch_prio</i> , <i>color</i> and <i>ecn</i> fields are ignored and the previous <i>switch_prio</i> , <i>color</i> and <i>ecn</i> values are preserved.	WO
	11:10	ecn	The new Packet ECN value for the relevant EXP value See Table 1614, “ECN bits Encoding,” on page 1502 for ECN values Note that the ECN is valid regardless of the trust state	RW
	9:8	color	The new packet color value for the relevant EXP value 0: GREEN 1: YELLOW 2: RED	RW
	3:0	switch_prio	The new Switch Priority value for the relevant EXP value.	RW

19.3.6 QPCR - QoS Policer Configuration Register

This register controls the per policer configuration.

Table 727 - QPCR - QoS Policer Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
								local_port								g		pid																00h																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
clear_counter	add_counter															color_aware	bytes		ir_units					type										mode	04h																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			

Table 728 - QPCR - QoS Policer Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port. Relevant only for per port policers No support for CPU port. No support for router port. Editing of this field is not allowed for a bounded policer. Reserved when Spectrum and for IB switches	Index
	15:14	g	0: per port policer configuration 2: global policer configuration 3: storm control policer configuration For IB Switches must be global	Index
	13:0	pid	Policer ID. Global policers are numbered from 0 to <i>cap_max_policers_global</i> - 1. Per port policers are numbered from 0 to <i>cap_max_policers_per_port</i> - 1. Storm policers are numbered from 0 to <i>cap_max_policers_storm_control</i> - 1 For Spectrum, a dual rate policer consumes 2 pids. A dual rate policer must have an even numbered pid.	Index

Table 728 - QPCR - QoS Policer Configuration Register Fields

Offset	Bits	Name	Description	Access
04h	31	clear_counter	Clear counters. Setting the <i>clear_counter</i> bit will reset the <i>violate_count</i> of the <i>pid</i> [Internal] IB switches: supported as internal for verification	OP
	30	add_counter	Add value to counters. The counter on the <i>pid</i> will be added with the value of <i>violate_count</i>	OP
	15	color_aware	0: meter is not color aware 1: meter is color aware Editing of this field is not allowed for a bounded policer Reserved storm control policers	RW
	14	bytes	0: meter based on packets/sec 1: meter based on bytes/sec This field is reserved when <i>mode</i> = 1 (packet sampling) Editing of this field is not allowed for a bounded policer Reserved when (support only packets/sec)	RW
	12	ir_units	0: <i>cir</i> and <i>eir</i> use units of 10 ⁶ bits/sec 1: <i>cir</i> and <i>eir</i> use units of 10 ³ bits/sec	OP
	9:8	type	policer type: 0: srTCM, single rate three colors marking. This mode is not supported by Spectrum 1: srDCM, single rate dual colors marking. This mode is not supported by SwitchX/-2 2: trTCM, dual rate three colors marking. This mode is not supported by SwitchX/-2 Editing of this field is not allowed for a bounded policer IB switches support only srDCM	RW
	1:0	mode	Operation Mode 0: policer 1: packet sampling Editing of this field is not allowed for a bounded policer	RW

Table 728 - QPCR - QoS Policer Configuration Register Fields

Offset	Bits	Name	Description	Access
08h	29:24	cbs	Committed burst size When <i>bytes</i> = 0: burst size is 2^{cbs} [packets] When <i>bytes</i> = 1: burst size is $2^{cbs} * 512$ [bits] SwitchX/-2: range is from 64Mbit to 8Gbit. Spectrum: When <i>bytes</i> = 1: Burst size is from <i>cir</i> * rate * 50uSec to 32Gbit, for <i>bytes</i> = 0 rate is 10^3 and for <i>bytes</i> = 1 rate is 10^3 or 10^6 according to <i>ir_units</i>. Also, <i>cbs</i> must always be larger than 3 Switch-IB/2 and Quantum: Range is 4.. 11 (16..2K Packets/Sec) When configured for packet sampling (<i>mode</i>=1), this field is used for range indicating the range of the uniform random. Range is in power of 2 packets. Random sampling will be uniform distribution in the following range: packet_rate – range : packet_rate + range TBD: do we want to expose the flexible range of Condor? not needing 2^{cbs}?	RW
	21:16	ebs	Excessive burst size When <i>bytes</i> = 0: burst size is 2^{ebs} [packets] When <i>bytes</i> = 1: burst size is $2^{ebs} * 512$ [bits] SwitchX/-2: range is from 64Mbit to 8Gbit. Spectrum: Burst size is from (<i>cir</i> (for single rate) or <i>eir</i> (for double rate)) * rate * 50uSec to 32Gbit, for <i>bytes</i> = 0 rate is 10^3 and for <i>bytes</i> = 1 rate is 10^3 or 10^6 according to <i>ir_units</i>. Also, <i>ebs</i> must always be larger than 3 Reserved when <i>type</i> is srDCM	RW

Table 728 - QPCR - QoS Policer Configuration Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	cir	Committed information rate. [Internal] For SwitchX/-2: Note that when configuring to low rates (below 500MB) will result with inaccurate results. For Spectrum: Resolution error is up to 1%. For bytes the units are according to <i>ir_units</i>. For packets metering CIR is in Packet/sec units. For SwitchX/-2: When configured for packet sampling, this field is used for <i>packet_rate</i> indicating the rate for packet sampling. Switch-IB/2: range: 6250 ... 100000 packet/sec. Resolution error is up to 10%. Quantum: supported values: N*64 packets/sec. Resolution error is up to 1%. Note: Reading will provide the exact value used by the HW [Internal] SwitchX/-2 are limited to resolution of 50mBit/Sec or 7000 Packets/Sec [Internal] SwitchX/-2 FW should generate the fill quanta 9 bit (in 64B units) and the fill interval in 2^N 8 bit i1 clocks. We only provide CIR here so that if internal clock changes, it will be easier to firmware to adjust. CIR = fill quanta/fill interval. [Internal] Spectrum resolution is 1024bit/Sec or 1 Packets/Sec. Spectrum has max 2Tbps or max 2Gpps	RW
10h	31:0	eir	Excess information rate. For Spectrum: Resolution error is up to 1%. Reading will provide the exact value used by the HW. For bytes the units are according to <i>ir_units</i>. For packets metering CIR is in Packet/sec units. Reserved when <i>type</i> is srDCM [Internal] Spectrum resolution is 1024bit/Sec or 1 Packets/Sec. Spectrum has max 2Tbps or max 2Gpps	RW
14h	3:0	exceed_action	Action for packets classified yellow by the policer (packets that exceed CBS but do not exceed EBS): 0: forward - Not supported by Spectrum. 1: discard - Not supported by Spectrum. 2: forward_and_set_color_to_yellow - Reserved when SwitchX/-2. Editing of this field is not allowed for a bounded policer. Reserved when <i>type</i> is srDCM	RW

Table 728 - QPCR - QoS Policer Configuration Register Fields

Offset	Bits	Name	Description	Access
18h	3:0	violate_action	Action for packets classified Red by the policer (e.g. packets that exceed CBS when <i>type</i> is srDCM or exceed EBS when <i>type</i> is trTCM): 0: forward - Not supported by Spectrum. 1: discard 2: forward_and_set_color_to_red - No support by SwitchX/-2	RW
20h	31:0	violate_count_high	Counts the number of times violate_actions happened on this pid [Internal] IB switches: supported as internal for verification. IB Switches: FW will map {swid,syndrome}->drop_counter in the same mapping as trap_group->pid. Note: the drop_counter counts both policer drops and tail-drops	RW
24h	31:0	violate_count_low		

[Internal] Spectrum policers use “statistical” method.

19.3.7 QPBR - QoS Policer Binding Register

This register controls the policer binding.

For SwitchX this register is used for binding of port policers and global policers.

For Spectrum this register is used for binding of Storm Control policers. When doing a get operation only 1 type of meter (uc, mc, bc, uuc, umc) can be set.

Table 729 - QPBR - QoS Policer Binding Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
op								local_port								g		pid												00h		
																										umc	uuc	bc	mc	uc	04h	
																															08h	
																															0C	

Table 730 - QPBR - QoS Policer Binding Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	op	Operation: 0: bind policer 1: update policer binding (status id bind) 2: unbind policer (For SwitchX/-2 only g and pid should be valid, for Spectrum the unbinding uses the same indexes as binding) For future: may add get command For Spectrum: “bind” and “update policer binding” do the same thing	RW
	23:16	local_port	local_port number. The policer is assigned to traffic coming from the port. SwitchX/-2: Note that one can bind a global policer to a port, and use the same global policer to more than a single port. Spectrum: Note that one can bind a Storm Control policer to a port.type, and use the same Storm Control policer to more than a single port. CPU port is not supported.	Index
	15	g	0: port policer and Storm Control policer configuration 1: global policer configuration Reserved when Spectrum	RW
	13:0	pid	Policer ID. Global policers are numbered from 0 to cap_max_policers_per_port-1 . Per port policers are numbered from 0 to cap_max_policers_global-1 . Storm Control policers are numbered from 0 to cap_max_policers_storm_control-1	RW
04h	4	umc	For Port or Storm Control: Meter ingress unknown multicast packets	Index
	3	uuc	For Port or Storm Control: Meter ingress unknown unicast packets	Index
	2	bc	For Port or Storm Control: Meter ingress broadcast packets	Index
	1	mc	For Port or Storm Control: Meter ingress multicast packets	Index
	0	uc	For Port or Storm Control: Meter ingress unicast packets	Index

19.3.8 QETCR - QoS ETS TC Configuration Register

This register controls the per port setting of the ETS per Traffic Class arbitration and shaping as well as the port's global shaping

For NIC devices supported only for Ethernet port when .ets == 1

Table 731 - QETCR - QoS ETS TC Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
operation_type	cap_local_admin	cap_remote_admin					local_port																									00h		
																																04h		
			tc_configuration[0]																															08h-0Ch
			tc_configuration[1]																															10h-14h
			...																															18h-3Ch
			tc_configuration[7]																															40h-44h
			global_configuration (Table 735, “ETS Global Configuration Register Layout”)																															48h-4Ch

Table 732 - QETCR - QoS ETS TC Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	operation_type	Operation type. 0x0: local_operative_parameters 0x1: local_admin_parameters - supported only when <i>cap_local_admin</i> ==1. 0x2: remote_admin_parameters - supported only when <i>cap_remote_admin</i> ==1. NIC only, Reserved for Switches	Index
	29	cap_local_admin	Indicates if <i>operation_type</i> ==local_admin_parameters is supported NIC only, Reserved for Switches	RO
	28	cap_remote_admin	Indicates if <i>operation_type</i> ==remote_admin_parameters is supported NIC only, Reserved for Switches	RO
00h	23:16	local_port	Local port number	Index
08h-44h	64	tc_configuration [8](Table 733, “ETS tcN Configuration Register Layout”)	Per-tclass configuration. For details, refer to Table 733, “ETS tcN Configuration Register Layout” .	
48h-4Ch	64	global_configuration (Table 735, “ETS Global Configuration Register Layout”)	Global configuration. For details, refer to Table 735, “ETS Global Configuration Register Layout” . Reserved for NICs	

Table 733 - ETS tcN Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
g	b	r													group																	00h
															max_bw_units																	04h

Table 734 - ETS tcN Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	31	g	Group update. Enables update of the group assigned to the TClass. If this bit is cleared, the <i>group</i> value will not change.	WO
	30	b	Bandwidth allocation update. Enables update of the ETS bandwidth allocation (with the <i>bw_allocation</i>). If this bit is cleared, the bandwidth allocation value will not change.	WO
	29	r	Rate limit update. Enables update of the rate limiter (with the <i>max_bw_units</i> , <i>max_bw_value</i>). If this bit is cleared, the rate limiter value will not change.	WO
	19:16	group	TClass Group assigned to a TClass. 0x8 value indicates no TClass Group is assigned to the TClass.	RW
	6:0	bw_allocation	For NICs: The weight assigned to the TClass within its TClass Group. The percentage of bandwidth guaranteed is calculated by the TClass weight divided by the sum of all weights of the TClasses in the group. Valid values are 1-100.	RW
04h	19:16	max_bw_units	Encoding of <i>max_bw_units</i> : 0x0: DISABLED - rate limiter disabled 0x3: HUNDREDS_MBPS - <i>max_bw_value</i> is in 100 Mbps units 0x4: GBPS - <i>max_bw_value</i> is in 1 Gbps units Note - Query response returns value in units that allow the most accurate number, not in the units configured using Set().	RW
	7:0	max_bw_value	The maximal bandwidth allowed for the use TClass Units according to <i>max_bw_units</i> Reserved when <i>max_bw_units</i> = 0	RW

Table 735 - ETS Global Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
		r																														00h
																																04h

Table 736 - ETS Global Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	29	r	Rate limit update. Enables update of the rate limiter (with the <i>max_bw_units</i> , <i>max_bw_value</i>). If this bit is cleared, the rate limiter value will not change.	WO
04h	19:16	max_bw_units	The maximal bandwidth allowed for the use for the port Encoding of <i>max_bw_units</i> : 0x0: DISABLED - rate limiter disabled 0x3: HUNDREDS_MBPS - <i>max_bw_value</i> is in 100 Mbps units 0x4: GBPS - <i>max_bw_value</i> is in 1 Gbps units Note - Query response will return value in units that allow the most accurate number, not in the units configured using Set().	RW
	7:0	max_bw_value		RW

<TBD> Port Rate Limiter - need to add

19.3.9 QEGCS - QoS ETS General Configuration Register

This register controls the per port global setting of the ETS arbiter:

Table 737 - QEGCS - QoS ETS General Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 738 - QEGCS - QoS ETS General Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index

Table 738 - QEGCS - QoS ETS General Configuration Register Fields

Offset	Bits	Name	Description	Access
04h	2	grp07_arb_sp	Arbitration inside groups 0..7: 0: DWRR 1: Strict Priority (default)	RW
	1	grp15_arb_sp	Arbitration inside group 15: 0: DWRR 1: Strict Priority (default) Note that if <i>grp07_arb_sp</i> is configured to Strict Priority <i>then-grp15_arb_sp</i> must be configured as Strict Priority as well	RW
	0	abr_glob_sp	Arbitration across groups: 0: DWRR 1: Strict Priority (default)	RW

19.3.10 QEEC - QoS ETS Element Configuration

This register configures the ETS elements.

Table 739 - QEEC - QoS ETS Element Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
												element_hi-erarchy													element_index								04h
																								next_element_index								08h	
mise		pips	pb	min_shaper_rate																								0Ch					
mase				max_shaper_rate																								10h					
pqe				phantom_queue_rate																								14h					
de																dwrtr									dwrr_weight								18h
											min_shaper_bs																max_shaper_bs				1Ch		
																										phantom_queue_bs							

Table 740 - QEEC - QoS ETS Element Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number CPU port supported.	Index
04h	19:16	element_hier- archy	The hierarchy of the element 0: Port 1: Group 2: Sub_Group 3: Traffic_Class	Index
	7:0	element_index	The index of the element in the hierarchy.	Index
08h	7:0	next_ele- ment_index	The index of the element in the lower hierarchy. reserved when <i>element_heirarchy</i> '0'	RW

Table 740 - QEEC - QoS ETS Element Configuration Register Fields

Offset	Bits	Name	Description	Access
0Ch	31	mise	Min shaper configuration enable. Enables configuration of <i>min_shaper_rate</i> , <i>min_shaper_bs</i> on this ETS element 0: disable 1: enable	RW
	29	ptps	PTP shaper 0: regular shaper mode 1: PTP oriented shaper Allowed only for hierarchy 0 Not supported for CPU port Note that <i>ptps</i> mode may effect the shaper rates of all hierarchies [Internal] Used as a workaround to Spectrum-1 PTP HW issue by providing port-rate based shaper which is slightly lower than port rate [Internal] SW should query shaper rates of all hierarchies when <i>ptps</i> is set or clear	RW
	28	pb	Packets or Bytes mode 0: bytes mode 1: packets mode Applies to both min shaper and max shaper For Spectrum packets mode is supported only for TClasses of CPU port. Reserved when <i>ptps</i> = 1	RW
	27:0	min_shaper_rate	Min shaper information rate For all hierarchies at Non-CPU port, and for port-hierarchy on CPU ports: bps mode, Units of 1000bps Granularity is 200mbps Error is up to 3% from configuration (for example: when rate is configured to 200mbps, actual rate may be 206mbps) Reading reveals the actual rate (e.g. 206 in the above example) Note: for rates lower than 10,000bps or lower than 100 packets per sec, error is up to 10% Reserved when <i>ptps</i> = 1	RW

Table 740 - QEEC - QoS ETS Element Configuration Register Fields

Offset	Bits	Name	Description	Access
10h	31	mase	Max shaper configuration enable. Enables configuration of <i>min_shaper_rate</i> , <i>min_shaper_bs</i> on this ETS element 0: disable 1: enable	RW
	27:0	max-shaper_rate	Max shaper information rate For all hierarchies at Non-CPU port, and for port-hierarchy on CPU ports: bps mode, Units of 1000bps Granularity is 200mbps Error is up to 3% from configuration (for example: when rate is configured to 200mbps, actual rate may be 206mbps) Reading reveals the actual rate (e.g. 206 in the above example) For Tclass of CPU ports: bps mode: - Granularity is 1000bps - Error is up to 1% from configuration packet per sec mode: - Granularity is 10 packet per sec - Error is up to 1% from configuration Note: for rates lower than 10,000bps or lower than 100 packets per sec, error is up to 10% Reserved when <i>ptps</i> = 1 [Internal] Spectrum HW also supports packet mode on the port itself	RW

Table 740 - QEEC - QoS ETS Element Configuration Register Fields

Offset	Bits	Name	Description	Access
14h	31	pqe	Phantom queue configuration enable. Enables configuration of the phantom queue rate 0: disable 1: enable Allowed only for hierarchy 0 Not supported for CPU port Reserved when <i>ptps</i> = 1	RW
	27:0	phantom_queue_rate	Phantom queue rate Units of 1000bps, bps only Granularity is 200mbps Error is up to 3% from configuration (for example: at 100Gbps, rate is configured to 200mbps, actual rate may be 206mbps) ECN marking is done when the rate is above <i>phantom_queue_rate</i> For Spectrum: When phantom queue is enabled then max shaper must be disabled. Disabling is done by setting the relevant rate to its max value. By default the phantom queue is disabled. Reserved when <i>ptps</i> = 1 [Internal] Spectrum: Bug CD #779580: Phantom Queue burst size is constant 4GByte, one would need up to 16MB buffer for phantom. So if we open phantom then we need FW WA to reset the phantom queue when the rate is lower than phantom rate, This also means no sub-group shaper and no group shaper since they become reset as well	RW
18h	31	de	DWRR configuration enable. Enables configuration of the <i>dwrr</i> and <i>dwrr_weight</i> . 0: disable 1: enable	RW
	15	dwrr	dwrr or strict priority on the link going down from the ETS element: 0: strict priority 1: DWRR	RW
	7:0	dwrr_weight	dwrr weight on the link going down from the ETS element: The percentage of bandwidth guaranteed to a ETS within its hierarchy. The sum of <i>dwrr_weight</i> value across all ETS elements within one hierarchy should equal 100. This field is reserved when <i>dwrr</i> is strict priority. [Internal] When weight is 0 there is no arbitration between the flows with weight of 0	RW

Table 740 - QEEC - QoS ETS Element Configuration Register Fields

Offset	Bits	Name	Description	Access
1Ch	21:16	min_shaper_bs	Min shaper Burst size Burst size is $2^{\text{min\_shaper\_bs}} * 512$ [bits] Range is: n..25 (from 2KB..2GB) Value of 0 is configured as n For Spectrum: n=5 Reserved when <i>ptps</i> = 1	RW
	5:0	max-shaper_bs	Max shaper Burst size Burst size is $2^{\text{max\_shaper\_bs}} * 512$ [bits] Range is: n..25 (from 2KB..2GB) Value of 0 is configured as n For Spectrum: n=5 Reserved when <i>ptps</i> = 1	RW

19.3.11 QPDP - QoS Port Default Priority

This register controls the port default Switch Priority and Color. The default Switch Priority and Color are used for frames where the trust state uses default values. All member ports of a LAG should be configured with the same default values.

For NIC devices supported only for Ethernet port when HCA_CAP.ets == 1

Table 741 - QPDP - QoS Port Default Priority Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 742 - QPDP - QoS Port Default Switch Priority Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port. Supported for data packets from CPU port.	Index
	15:8	sub_port	Virtual port within the physical port. 0 when virtual ports are not enabled on the port.	Index
04h	9:8	color	Default port color (default=0) 0x0: GREEN 0x1: YELLOW 0x2: RED For Switch only, Reserved for NICs	RW
	3:0	switch_prio	Default port Switch Priority (default 0) For NIC devices only 8 priorities are supported	RW

19.3.12 QPDPC - QoS Port Default PCP

This register controls the port default PCP and DEI. These PCP and DEI are used for untagged frames. All member ports of a LAG should be configured with the same default values.

Table 743 - QPDPC - QoS Port Default PCP Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port																								00h
																							dei							pcp		04h

Table 744 - QPDPC - QoS Port Default PCP Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port. No support for CPU port.	Index
	15:8	sub_port	Virtual port within the physical port. 0 when virtual ports are not enabled on the port.	Index
04h	8	dei	Default port DEI (default 0) when	RW
	2:0	pcp	Default port PCP (default 0)	RW

19.3.13 QPRT - QoS Priority Regeneration Table

This register configures the table that maps ingress frame priority (PCP,DEI) to internal Switch Priority¹. All member ports of a LAG should be configured with the same priority regeneration values.

[Internal] Spectrum: This is actually the Priority to Switch Priority mapping. The current name is according to the IEEE function

Table 745 - QPRT - QoS Priority Regeneration Table Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 746 - QPRT - QoS Priority Regeneration Table Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port.	Index
	11	dei	Packet's DEI value. when	Index
	10:8	pcp	Packet's PCP value	Index
04h	9:8	color	Packet Color. 0: GREEN 1: YELLOW 2: RED For Switch only, Reserved for NICs	RW
	3:0	rprio	For SwitchX/-2: Regeneration Priority. For Spectrum: Switch Priority The default value for the table is: PCP i ->Regeneration Priority i For NIC devices only 8 priorities are supported	RW

1. In SwitchX/-2 the internal switch priority is used for rewrite of the packet's PCP. In Spectrum, software controls the rewrite function.

19.3.14 QSPTC - QoS Stacking Port TClass Table

The traffic class for packets that enter on stacking link and exit on stacking link can be altered. This register controls the stacking TClass regeneration for intermediate devices.

Table 747 - QSPTC - QoS Stacking Port TClass Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset											
local_iport								local_eport												itclass																00h							
																																				tclass				04h			

Table 748 - QSPTC - QoS Stacking Port TClass Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	local_iport	Local ingress port number (must be stacking port)	Index
	23:16	local_eport	Local egress port number (must be stacking port)	Index
	10:8	itclass	Received TClass	Index
04h	2:0	tclass	Regenerated stacking TClass for received packets on stacking port with TClass The default value for the Stacking Port TClass Regeneration Table is tclass[itclass=i]=i.	RW

19.3.15 QTCT - QoS Switch Traffic Class Table

This register controls the port Traffic Class Table. The table controls the mapping between the packet switch priority and the traffic class on the transmit port. All member ports of a LAG should be configured to the same traffic class.

For Switches: The Switch Priority to Traffic Class has a mode of MC aware, see register QTCTM [Section 19.3.16, “QTCTM - QoS Switch Traffic Class Table is MultiCast aware,” on page 965.](#)

For NIC devices supported only for Ethernet port when HCA_CAP.ets == 1, all virtual ports share the same configuration with the physical port.

Table 749 - QTCT - QoS Switch Traffic Class Table Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
operation_type	cap_local_admin	cap_remote_admin						local_port																								00h
																																04h

Table 750 - QTCT - QoS Switch Traffic Class Table Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	operation_type	Operation type. 0x0: local_operative_parameters 0x1: local_admin_parameters - Supported only when cap_local_admin==1 0x2: remote_admin_parameters- Supported only when cap_remote_admin==1 when	Index
	29	cap_local_admin	Indicates if operation_type==local_admin_parameters is supported when	RO
	28	cap_remote_admin	Indicates if operation_type==remote_admin_parameters is supported. when	RO
	23:16	local_port	Local port number (transmit port)	Index
	15:8	sub_port	Virtual port within the physical port. 0 when virtual ports are not enabled on the port.	Index
	3:0	switch_prio	Switch Priority	Index

Table 750 - QTCT - QoS Switch Traffic Class Table Register Fields

Offset	Bits	Name	Description	Access
04h	3:0	tclass	Traffic Class Default values are: - switch_prio=0: tclass=1 - switch_prio=1: tclass=0 - switch_prio>1: tclass = switch_prio When MC aware mode is enabled, setting values greater or equal to (<i>cap_max_tclass_data</i>)/2) is not allowed	RW

19.3.16 QTCTM - QoS Switch Traffic Class Table is MultiCast aware

This register configured if the Switch Priority to Traffic Class mapping is based on Multicast packet indication. Switch Priority to Traffic Class mapping is based on Multicast packet indication then multicast packets will get a Traffic Class that is plus (*cap_max_tclass_data/2*) the value configured by QTCT.

By default, Switch Priority to Traffic Class mapping is not based on Multicast packet indication.

Table 751 - QTCTM - QoS Switch Traffic Class Table is Multicast aware Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 752 - QTCTM - QoS Switch Traffic Class Table is Multicast aware Register Layout

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port No support for CPU port	Index
04h	0	mc	Multicast Mode 0: Switch Priority to Traffic Class mapping is not based on Multicast packet indication (default) 1: Switch Priority to Traffic Class mapping is based on Multicast packet indication	RW

19.3.17 QSTCT - QoS Switch Stacking TClass Table

This register selects the stacking Traffic Class on ingress devices for packets that are forwarded into a stacking interface. The table controls the mapping per switch partition between the packet switch priority and the stacking traffic class. Unicast and multicast packets can have different mapping. **[Internal] This is a single table per device**

Table 753 - QSTCT - QoS Switch Stacking TClass Table Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
swid																				switch_prio												00h
																																04h
																																08h
																																0Ch

Table 754 - QSTCT - QoS Switch Stacking TClass Table Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid	Switch partition ID	Index
	11:8	switch_prio	Ingress Switch Priority	Index
04h	2:0	utclass	Stacking TClass used for unicast packets with priority switch_prio. Default values are: - utclass[switch_prio=0]=1, - utclass[switch_prio=1]=0, - utclass[switch_prio=<i>]=i (for i>1).	RW
08h	2:0	mtclass	Stacking TClass used for multicast packets with priority prio. Default values are: - mtclass[switch_prio=0]=1, - mtclass[switch_prio=1]=0, - mtclass[switch_prio=<i>]=i (for i>1).	RW

19.3.18 QSPIP - QoS Switch Priority to IEEE Priority Mapping Table

This register controls the mapping from Switch Priority to IEEE Priority. This register configures all local ports (receive and transmit) except CPU port and router port.

Table 755 - QSPIP - QoS Switch Priority to IEEE Priority Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																												switch_prio				00h	
																													ieee_prio				04h

Table 756 - QSPIP - QoS Switch Priority to IEEE Priority Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	switch_prio	Switch Priority entry form mapping to IEEE Priority	Index
04h	2:0	ieee_prio	The new IEEE Priority value for the relevant Switch Priority value. default: for i=[0..7]: switch_prio i: ieee_prio i for i=[8..15]: switch_prio i: ieee_prio (i-8) Note: Different switch_prio values assigned to a given buffer (see PPTB) may be assigned to different ieee_prio values.	RW

19.3.19 QSPCP - QoS Switch Priority to Counting Priority Mapping Table

This register controls the mapping from Switch Priority to Counting Priority. This register configures all local ports (receive and transmit) except CPU port and router port.

Table 757 - QSPCP - QoS Switch Priority to Counting Priority Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 758 - QSPCP - QoS Switch Priority to Counting Priority Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port No support for CPU port No support for router port [Internal] for legacy reasons, when <i>local_port</i> is '0' then the configuration is global and reading is done from local_port 1	Index
	3:0	switch_prio	Switch Priority entry form mapping to Counting Priority. [Internal] For Spectrum <i>switch_prio</i> is from 0 to 14	Index
04h	2:0	rx_counting_prio	The Rx Counting Priority value for the relevant Switch Priority value. default: for i=[0..7]: switch_prio i: rx_counting_prio i for i=[8..15]: switch_prio i: rx_counting_prio (i-8)	RW
08h	2:0	tx_counting_prio	The Tx Counting Priority value for the relevant Switch Priority value. default: for i=[0..7]: switch_prio i: tx_counting_prio i for i=[8..15]: switch_prio i: tx_counting_prio (i-8)	RW

19.3.20 QRWE - QoS ReWrite Enable

This register configured the rewrite enable per receive port.

Table 759 - QRWE - QoS ReWrite Enable Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port																								00h
																													exp	dscp	pcp	04h

Table 760 - QRWE - QoS ReWrite Enable Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port Supported for CPU port No support for router port	Index
04h	2	exp	exp_rewrite_enable: 0: disable EXP rewrite, this is the default 1: enable EXP rewrite	RW
	1	dscp	dscp_rewrite_enable: 0: disable DSCP rewrite, this is the default 1: enable DSCP rewrite	RW
	0	pcp	pcp_rewrite_enable: 0: disable PCP and DEI rewrite, this is the default 1: enable PCP and DEI rewrite	RW

19.3.21 QPEM - QoS Priority to EXP Mapping Register

This register configures the mapping from switch priority to MPLS EXP field for MPLS labeled packets on the transmit port.

Table 761 - QPEM - QoS Priority to EXP Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
switch_prio[0]																																04h-10h	
switch_prio[1]																																14h-20h	

Table 761 - QPDM - QoS Priority to EXP Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
...																												..				
switch_prio[15]																												F4h-100h				

Table 762 - QPDM - QoS Priority to EXP Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port. No support for CPU port. No support for router port.	Index
04h-100h	128	switch_prio[16]T able 763, “switch_prio<i> > Layout”	The EXP mapping for Switch Priority <i>i</i> , See Table 763 , “switch_prio<i> > Layout” Switch Prio range is from 0 to <i>cap_max_switch_priorities</i> -1	

Table 763 - switch_prio<i> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ecn[0]																												00h				
ecn[1]																												04h				
...																												08h				
ecn[3]																												0Ch				

Table 764 - switch_prio<i> Fields

Offset	Bits	Name	Description	Access
00h-0Ch	31:0	ecn[4]Table 7 65, “ecn<j> Layout”	The EXP mapping for ECN _{<i>j</i>} , see Table 765 , “ecn<j> Layout”	

Table 765 - ecn<j> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
color0								color1								color2																00h

Table 766 - ecn<j> Fields

Offset	Bits	Name	Description	Access
00h	31:24	color0	The EXP mapping for color k , where $k=0..2$, see Table 767 , “color<k>Layout,” on page 971	
	23:16	color1		
	15:8	color2		

Table 767 - color<k>Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								e						exp		00h

Table 768 - color<k> Fields

Offset	Bits	Name	Description	Access
00h	7	e	Enable update of the specific entry	WO
	2:0	exp	EXP field in the outer label of the packet Reserved when $e = 0$	RW

19.3.22 QPDSM - QoS Priority to DSCP Mapping Register

This register configures the mapping from switch priority to DSCP field for IP packets on the transmit port.

Table 769 - QPDSM - QoS Priority to DSCP Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
switch_prio[0]																																04h	

Table 769 - QPDSM - QoS Priority to DSCP Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
switch_prio[1]																																08h
...																																..
switch_prio[15]																																40h

Table 770 - QPDSM - QoS Priority to DSCP Mapping Register Fields

Offset	Bits	Name	Description	Access
0h	23:16	local_port	Local port. No support for CPU port.	Index
04h-40h	31:0	switch_prio[16] Table 771, "switch_prio<j> Layout"	The DSCP mapping for Switch Priority <i>i</i> , See Table 771 , "switch_prio<j> Layout" Switch Prio range is from 0 to <i>cap_max_switch_priorities</i> -1	

Table 771 - switch_prio<j> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
color0								color1								color2																00h

Table 772 - prio<j> Fields

Offset	Bits	Name	Description	Access
00h	31:24	color0	The DSCP mapping for color <i>k</i> , where <i>k</i> =0..2, see Table 773 , "color<k>Layout"	
	23:16	color1		
	15:8	color2		

Table 773 - color<k>Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								e		dscp					00h	

Table 774 - color<k> Fields

Offset	Bits	Name	Description	Access
00h	7	e	Enable update of the specific entry	WO
	5:0	dscp	EXP field in the outer label of the packet Reserved when $e = 0$	RW

19.3.23 QPPM - QoS Priority to PCP Mapping Register

This register configures the mapping from switch priority to PCP and DEI fields for tagged packets on the transmit port.

Table 775 - QPPM - QoS Priority to PCP Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
local_port																																04h
																switch_prio[0]																08h
																switch_prio[1]																..
																...																40h
																switch_prio[15]																

Table 776 - QPPM - QoS Priority to PCP Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port. No support for CPU port.	Index
04h-40h	31:0	switch_prio[16] Table 777, "switch_prio<j> Layout"	The PCP mapping for Switch Priority i , See Table 777 , "switch_prio<j> Layout" Switch Prio range is from 0 to $cap\_max\_switch\_priorities-1$	

Table 777 - switch_prio<j> Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
color0								color1								color2																00h

Table 778 - switch_prio<j> Fields

Offset	Bits	Name	Description	Access
00h	31:24	color0	The PCP and DEI mapping for color k , where $k=0..2$, see Table 779, "color<k>Layout"	
	23:16	color1		
	15:8	color2		

Table 779 - color<k>Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								e			dei				pcp	00h

Table 780 - color<k> Fields

Offset	Bits	Name	Description	Access
00h	7	e	Enable update of the specific entry	WO
	4	dei	DEI field in the outer VLAN tag of the packet Reserved when $e = 0$	RW
	2:0	pcp	PCP field in the outer VLAN tag of the packet Reserved when $e = 0$	RW

19.3.24 QSL - QoS Switch Lifetime Limit Register

Table 781 - QSL - QoS Switch Lifetime Limit Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																															sll_time	04h

Table 782 - QSLL - QoS Switch Lifetime Limit Register Fields

Offset	Bits	Name	Description	Access
04h	4:0	sll_time	SLL Time $SLL\ Time = 4.096\mu Sec * 2^{sll\_time}$ Default is 20 (4.3Sec) Range is 3 ..30 Value of 31 means infinite (no SLL)	RW

19.3.25 QHLL - QoS Head of queue Lifetime Limit Register

[Internal] Note for Spectrum-1: when changing *hll_time* or *stall_cnt* then next stall state may start earlier (by up to 0Sec) or later by up to $32 * hll\_time$. Fixed for Quantum and Spectrum-2.

Table 783 - QHLL - QoS Head of queue Lifetime Limit Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
stall_en																																08h

Table 784 - QHLL - QoS Head of queue Lifetime Limit Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port	Index
04h	4:0	hll_time	HLL Time HLL Time = $4.096\mu\text{Sec} * 2^{hll\_time}$ Range is 0 ..19 Values of 20-31 means infinite (no HLL)	RW
08h	31	stall_en	Stall enable 0: Stall disabled 1: Stall enabled	RW
	2:0	stall_cnt	Stall count. Number of HLL discarded packets required to enter the Stall-State. Range 1..7 Reserved when <i>stall_en</i> = 0	RW

19.3.26 QCAM - QoS Capabilities Mask Register

This register is not used by Switch devices

The QCAM register reports the device supported QoS registers and features.

NOTE: For HCA devices, this register is supported only when QCAM == 1.

Table 785 - QCAM - QoS Capabilities Mask Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-14h
																																18h-24h
																																28h-34h
																																38h-44h

Table 786 - QCAM - QoS Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	feature_group	Feature list mask index 0: Enhanced_features	Index
	7:0	access_reg_group	Access Register ID groups 0: First_128_REG_ID	Index
08h-14h	128	qos_access_reg_cap_mask	Supported QoS access register bitmasks. Based on access_reg_group index. When bit is set, the register is supported in the device. For example: Bit 1: QCAP (0x4001) Bit 2: QPTS (0x4002) Bit 3: QDPM (0x4003)	RO
28h-34h	128	qos_feature_cap_mask	Supported QoS enhanced features based on feature_group index. When bit is set, the feature is supported in the device Bit 0: QPTS_TRUST_BOTH - trust_both is supported in QPTS register	RO

19.3.27 QPSC - QoS PTP Shaper Configuration Register

[Internal]

This register is not used by NIC devices

The QPSC allows advanced configuration of the shapers when QEEC.ptps=1

NOTE: For HCA devices, this register is supported only when HCA_CAP.QCAM == 1.

Table 787 - QPSC - QoS PTP Shaper Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												port_speed				00h
												shapers_ - time_exp																shaper_ - time_mantisa				04h
																												shaper_inc				08h
																												shaper_bs				0Ch
disc we																										port_to_shaper_credits				10h		
																																14h-1Ch
ing_timestamp_inc																																20h
egr_timestamp_inc																																24h

Table 788 - QPSC - QoS PTP Shaper Configuration Register Layout

Offset	Bits	Name	Description	Access
00h	3:0	port_speed	Port speed: 0: 100 Mbps 1: 1 Gbps 2: 10 Gbps 3: 25 Gbps 4: 40 Gbps 5: 50 Gbps 6: 100 Gbps	Index
04h	19:16	shapers_time_exp	The base-time-interval for updating the shapers tokens (for all hierarchies) $shaper\_update\_rate = 2^{shapers\_time\_exp} * (1 + shaper\_time\_mantisa) * 32nSec$	RW
	4:0	shaper_time_mantisa	$shaper\_rate = 64bit * shaper\_inc / shaper\_update\_rate$ [Internal] FW: dqs_4x[].split[].counters_table.timer_for_shapers_exp dqs_4x[].split[].flow_ctrl_table.shapers.shaper_port_green.shaper_inc_interval	RW

Table 788 - QPSC - QoS PTP Shaper Configuration Register Layout

Offset	Bits	Name	Description	Access
08h	4:0	shaper_inc	Number of tokens added to shaper on each update Units of 8B [Internal] FW: dqs_4x[].split[].flow_c- trl_table.shapers.shaper_port_green.shaper_inc_tokens Value of 0 means always block	RW
0Ch	5:0	shaper_bs	Max shaper Burst size Burst size is $2^{\text{max\_shaper\_bs}} \times 512$ [bits] Range is: 5..25 (from 2KB..2GB) Range is: 0..25 (from 64B..2GB) [Internal] same as QEEC register	RW
10h	31	ptsc_we	Write enable to <i>port_to_shaper_credits</i>	WO
	7:0	port_to_shaper_credits	For split ports: range 1..57 For non-split ports: range 1..112 Written only when <i>ptsc_we</i> is set [Internal] FW: dqs_4x[].split[].g_credit_llu_reflection	RW
20h	31:0	ing_time-stamp_inc	Ingress timestamp increment 2's complement The timestamp of MTPPTR at ingress will be incremented by this value. Global value for all ports Same units as used by MTPPTR Default TBD	RW
24h	31:0	egr_time-stamp_inc	Egress timestamp increment 2's complement The timestamp of MTPPTR at egress will be incremented by this value. Global value for all ports Same units as used by MTPPTR Default TBD	RW

20 Advanced I/O Operation Flows

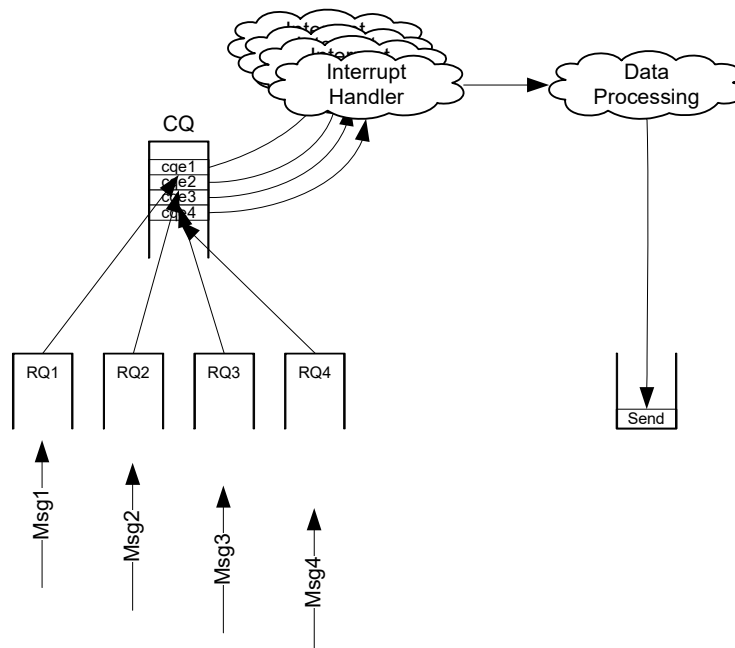
The device can be programmed to execute I/O operation flows that involve synchronization of I/O operations' on different I/O channels as well as conditional execution based on arithmetic operations on memory-resident variables. This capability enables to program complex I/O operation flows with a single function call, hereby significantly reducing overhead associated with I/O processing.

20.1 Cross-channel I/O Operations

This section describes the device capabilities to synchronize/serialize execution of I/O operation on different WQs without any intervention from the host software.

Consider complex I/O operation that processes data received from multiple sources and forwards the result to a single destination. This scenario is illustrated in [Figure 93](#).

Figure 93: Forwarding Multiple Messages to a Single Destination



The figure shows the case where data received from 4 different sources is forwarded as a single message after some processing.

In order to implement transaction shown in [Figure 93](#), host software needs to receive completions from each one of the RQs (arriving in arbitrary order), process the data after last message arrival and only then post WR on the SQ to send the combined data to its destination.

Execution of such an operation generates multiple interrupts at an unpredictable time, interfering with host OS scheduler creating what is called “OS noise” - interrupt-related overhead of receive message processing. On the other hand, in most cases the data processing operation is a fairly simple one and interrupt and context switch overhead exceeds by far time required by the host to process data. Thus, such an operation inherently imposes inefficiency on the host.

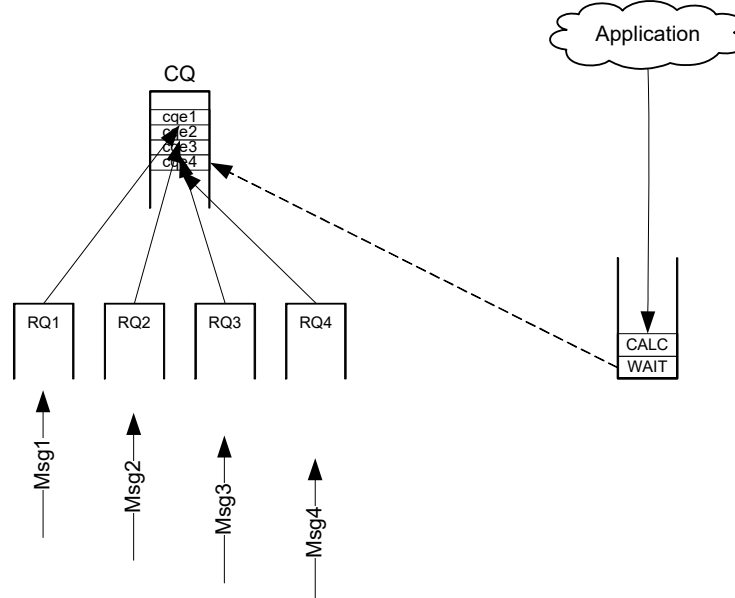
The device implements capability to execute complex I/O operations across multiple I/O channels - synchronize execution of I/O operations on one I/O channel with execution of I/O operations on another channel(s) as well as several basic - albeit powerful - data processing operations, covering the most common cases of data forwarding.

Cross-channel operations support is indicated by the `HCA_CAP.cd`. The device provides basic primitives that can be posted on a flow interface SQ configured to support advanced cross-channel operations. Those primitives are posted on a flow interface designated as a “sync master queue”. Some of the primitives apply to other flow interfaces; those are referred to as “sync slave queues”. The device supports the following primitives:

- **WAIT on CQ WR** - This WR holds execution of subsequent requests on that queue until *wait* condition is met. *Wait* condition is a value of producer index of a CQ. As long as producer index does not reach value specified by WAIT WR, execution of subsequent WQEs on this SQ is suspended. Once producer index exceeds value specified in WAIT WR, the execution on that SQ is resumed.
- **send_EN WR** - This WR specifies value of Producer Index on the controlled flow interface SQ, specified in the WR. All WQEs up to this Producer Index are enabled for execution. Software must assure that descriptors are posted to the respective SQ prior to execution of `send_EN` command.
- **RECEIVE_EN WR** - Same as `send_EN` but related to a RQ. Does not work with streaming queue RQs.
- **CALC WR** - This WR executes compute operation on memory-resident arguments specified in WR. Result of this calculation is sent to the remote peer on the network (or to the originating host via loopback). CALC WR can be posted on either Datagram or Connected SQ.

Armed with these primitives, the sequence of operations shown in [Figure 93](#) can be executed in hardware without any asynchronous interrupts to the host, and is illustrated in [Figure 94](#).

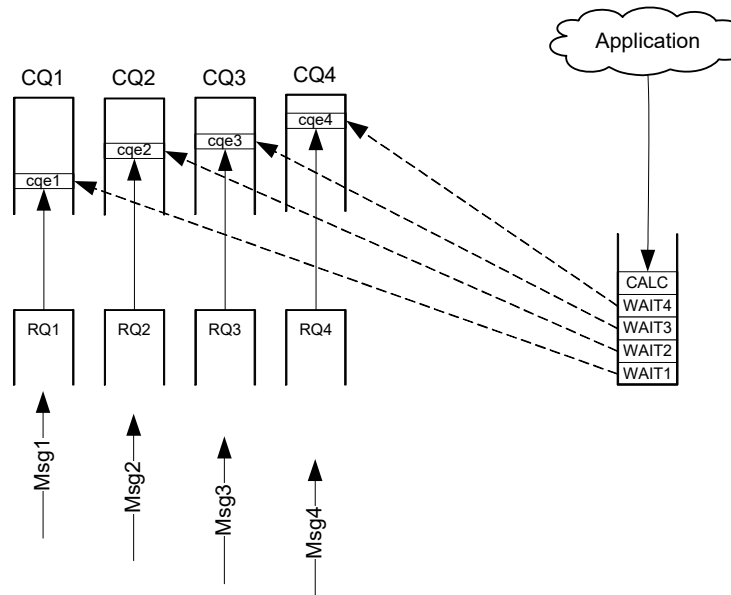
Figure 94: Offloaded Forwarding of Multiple Messages to a Single Destination



An application posts WAIT WR followed by CALC request. Execution of the CALC request is held off until WAIT condition is met - which is report of completion of fourth message on CQ. Note that posting messages on SQ does not require any synchronization with receive execution - send messages could be posted before or after all receive operations are completed. In former case WAIT instruction will block execution of CALC operation until all receive operations are completed. In latter case, the condition of WAIT operation would be met right away and execution on SQ will proceed.

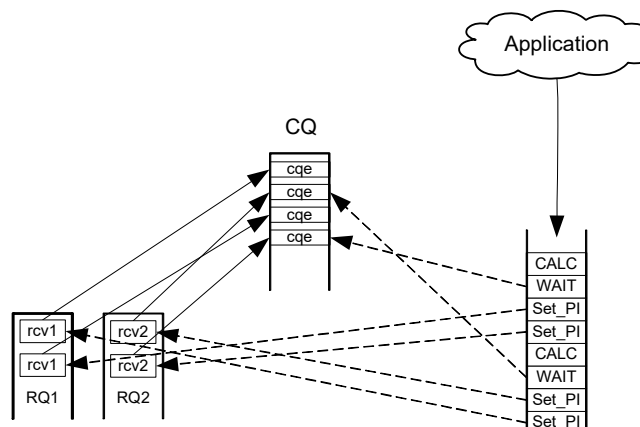
Multiple WAIT commands can be used to synchronize completion reports on multiple CQs. [Figure 95](#) demonstrates a case where each RQ reports completion to a different CQ. WAIT commands are used to synchronize between receive and send execution.

Figure 95: Synchronization Across Multiple Completion Queues



A combination of WAIT and RECEIVE_EN commands can synchronize receive operation with I/O operations on other queues. Extending the example of [Figure 94](#), consider a case where it is desired to reuse a single receive buffer on each RQ. [Figure 96](#) demonstrates pipelined operation: software posts multiple receive descriptors to the RQ(s) specifying the same receive buffer. Software then posts RECEIVE_EN I/O operations on the SQ that set WQE_max on each RQ synchronously with processing of received data. Remote peer that sends the data will be backed off by RnR mechanism at transport layer, avoiding a need to notify sender about responder readiness. As a result, same receive buffer on each RQ is re-used for data receive and it is “re-cycled” over to the next receive operation after processing of data was completed.

Figure 96: Pipelined Data Forwarding



To enable a flow interface support for processing cross-channel primitives, designate it as a “sync master queue” by setting the <cd_master> field in the QP context when creating the QP.

To enable a flow interface to respect send_EN and RECEIVE_EN requests, designate it as a “sync slave queue” by setting the cd_slave_send and cd_slave_receive fields in the QP context when creating the QP.

20.1.1 Using Cross Channel Operations for MPI Collective Offload

Cross-channel synchronization operations along with CALC operations can be used to implement MPI collectives’ operations within the adapter, reducing significantly the amount of interrupts and interference with the host processor, hereby enabling to overlap collectives’ execution with computing process.

20.1.2 Cross-channel Operation Work Request Format

Cross-channel operation is specified by OPCODE field value of WAIT, send_EN, RECEIVE_EN or CALC in the WQE control segment ([Section 8.9.4.1.1, “Ctrl Segment,” on page 312](#)). WAIT on CQ and WAIT on QP are distinguished in the OPC_MODE field.

20.1.2.1 CALC Work Request

Integer and Floating Point calculation operations are specified in the OPC_MOD field of control segment.

CALC result is interpreted as Send or Write opcode and the calculation results are sent to the destination of this request.

CALC can have up to 16 arguments.

All calculations (integer and floating point) are 64-bit precision.

Data segment of the CALC WQE is a pointer and is the same as for any Send or RDMA Write WQE.

MAXLOC operations have an additional 8 byte tag (node rank ID). The vbuffer of MAXLOC operation is shown on [Table 789](#). For other operation, the tag field is reserved.

Table 789 - CALC Buffer for MAXLOC Operation

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
Data[63:32]																															00h
Data[31:0]																															04h
tag[63:32]																															08h
tag[31:0]																															0Ch

Note: MPI_MAX is implemented by MPI_MAXLOC when the tag fields are not in use.

Note: MPI_MIN and MPI_MINLOC are implemented by max when the data is inverted before posting the operation and after receiving the results.

Note: MPI LAND, LOR and LXOR can be implemented by band, bor and bxor within the fabric. At the end node, SW will calculate the logic result.

20.1.2.2 WAIT, SEND_EN and RECEIVE_EN Work Requests

WAIT, send_EN and RECEIVE_EN WRs are specified by appropriate value of OPCODE field in the WQE control format and include *Sync QP* segment specified in [Table 790](#).

Table 790 - Sync QP Segment

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										
																												00h			
																												04h			
												max_index																08h			
												qpn_cqn																0Ch			

Fields of *Sync QP* segment are specified in [Table 791](#).

Table 791 - Sync QP Segment Fields

Offset	Bits	Field	Description	Access
08h	23:0	max_index	For send_EN and RECEIVE_EN operations, this is the rq_wqe_max / sq_wqe_max to be written to the sync slave QP. The sync slave QP will resume WQE processing up to this index. Only 16 lsb's of this field are considered. For WAIT operation, this is the CQ producer index, SQ consumer index or RQ consumer index. The sync master QP will suspend its WQE processing until the producer/consumer index of the CQ/QP specified in the <qpn_cqn> field reaches this index.	
0Ch	23:0	qpn_cqn	QP / CQ number on which the operation is executed	

Hardware validates that Extended Protection Domain {GVMI, PD} of SQ WQE is posted on matches Extended Protection Domain of WQ/CQ it is referred to and executes the command. If mismatch encountered in Extended Protection Domain, the WR completes with error with all circumstances.

20.1.3 Completion and Error Reporting

When a list of control WQEs is generated by the driver, the driver may mark any one of the WQEs to report completion upon. For example, for a sequence of RECEIVE_EN, WAIT and send_EN, WAIT WQEs, the driver may mark the first WAIT WQE for completion (for lower latency), or the second WAIT WQE (so completion of send is assured).

WQEs can be completed with error on both the controlling flow interface and the controlled flow interface. In both cases, the control WQE will be completed with error indication.

20.2 Vector-Calc Operations

Note: This feature is supported only when `HCA_CAP.vector_calc==1`.

Vector-calc is an extension to the CD capability that takes as input several vectors of equal length and combines them with a single operation type to produce result vectors which are sent to the destination. However, this is introduced as a feature that is independent of the CD capabilities. For all but the XOR operation, it is possible to produce only one output vector.

The result of the calc operation is interpreted as a send or write operation where `opmod==0xff` as described in [Table 44, “OPC_MOD Field Encoding,” on page 316](#) with no Data Segment. Instead of Data Segment, a Vector Calc segment is used (See [Table 77, “Vector Calc Segment Format,” on page 336](#)).

20.3 Tag Matching and Rendezvous Offload

20.3.1 Overview

Note: This section is preliminary and subject to change.

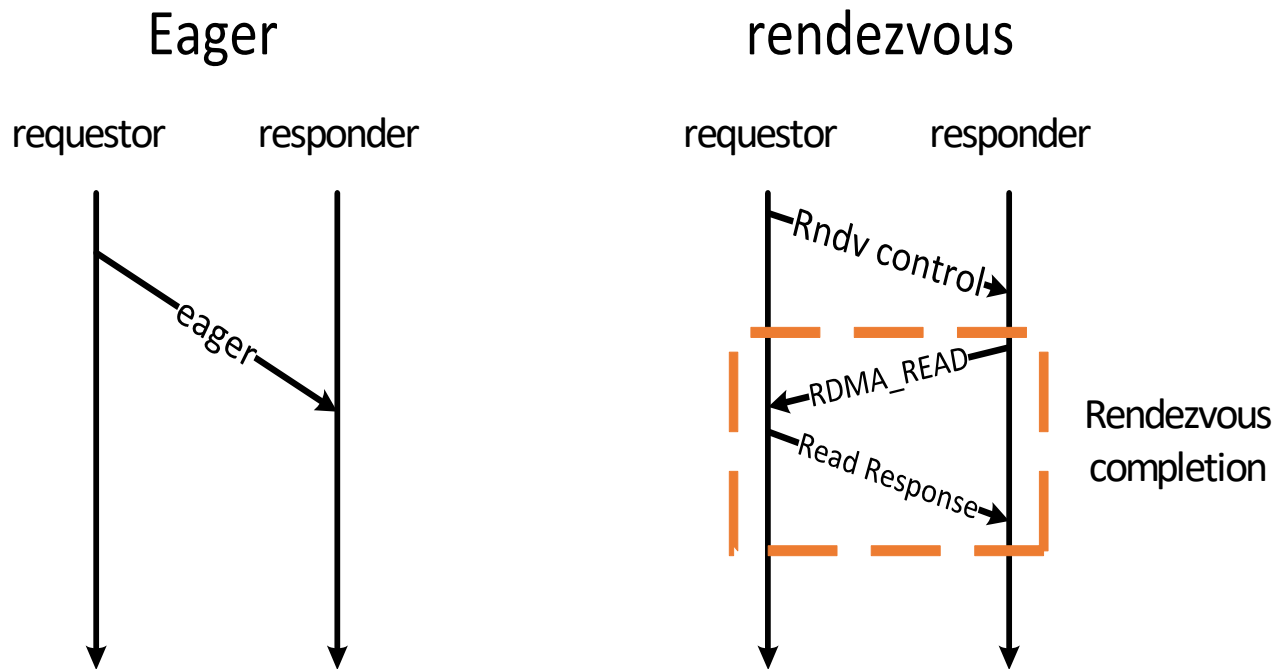
The support for tag matching offload is indicated by ***tag_matching*** field in HCA_CAP (see [Table 1571, “HCA Capabilities Layout,” on page 1671](#)).

In tag matching, SW holds a list of matching entries called matching list. Each matching entry contains a tag and a pointer to an application buffer. The matching list is used to steer arriving messages to a specific buffer according to the message tag. The action of traversing the matching list and finding the matching entry is called tag matching.

There are two common protocols for tag matching:

- eager - on which data is passed with the message.
- rendezvous - on which the requester sends a small control message (called rndv control) notifying its intent to pass data. The responder will complete the rendezvous by initiating RDMA operation to obtain the data. When the RDMA operation completes, the target will indicate to the client on the end of operation by sending a FIN notification.

Figure 97: Eager and Rendezvous Flows



Tag matching offload enables SW to push the head of the matching list to the HCA in order to preform HW tag matching. If the matching entry is found, the message will be scattered directly to the application buffer. This message will be called expected. If the device does not find the matching entry, it will pass the message to SW. This message will be called unexpected. In that case, SW will complete the tag matching operation on the rest of the list.

In order to use tag matching offload target, SW will need to:

- Create XRQ with tag matching topology (See [Section 8.21, “XRQ Context,”](#) on page 509).
- Connect QPs to it (in the same way it connects QPs to SRQ)
- Push the head of the matching list to HW (using append list operation)

Naturally, race condition is possible between passing of matching entry to device while its matching message arrives from the network. Therefore, it is possible that the matching entry will be passed to the device but its corresponding message will arrive to SW as unexpected. In that case, SW will need to remove the matching entry from the device using remove list operation. In addition, XRQ must know to ignore this matching entry in tag matching operation until it is removed. In order for XRQ to ignore this matching entry, XRQ contains **sw_phase_cnt** and **hw_phase_cnt**. When counters are not equal, the new matching entry appends will not take part in the tag matching operation. When counters are equal, all matching entries in the matching list will take part in the tag matching operation. **hw_phase_cnt** field is increased by 1 every time the device passes unexpected messages to SW. **sw_phase_cnt** is updated when SW updates it using list operation WQE.

SW should keep a local value of *sw_phase_cnt* and increase it by 1 every time it finishes handling unexpected messages. SW should update its local *sw_phase_cnt* to HCA using update counter list operation. Note that tag matching is performed only for Send/Send with immediate/Send with invalidate opcodes.

In addition, the device supports rendezvous offload. In rendezvous offload, the device identifies rendezvous control messages and completes the rendezvous by initiating RDMA READ operation from the remote side. When the RDMA_READ completes, the device will generate a FIN message indicating the end of the operation. FIN message is a regular IB_Send operation, sent to the same target as the RDMA_READ. It takes the RNDV control TMH as a payload, clears the TMH has_tag field and modifies TMH.protocol to be *FIN*. The support for rndv offload and the maximal size for rendezvous control message supported according to transport service is indicated by *rndv_offload_dc*, *rndv_offload_rc* and *log_max_rndv_payload_size* fields in HCA _CAP (See [Table 1571, “HCA Capabilities Layout,” on page 1671](#)). rndv offload target SW will need to:

- Create XRQ with offload=*rndv* and topology=tag_matching.
- Connect QPs to it

The transport service on which the rendezvous completion is executed is according to the transport service of the QP which addresses the XRQ:

- In case QP transport is RC – *in case the QP will receive rndv messages* the QP send side will be ~~is~~ in device ownership and is used to complete the rndv. SW should create the QP, connect it to XRQ and pass it to RTS state. It must not post any WQE to QP send side. SW is allowed to post send WQEs to the QP which is connected to the XRQ with rndv offload but does not receive any rndv messages.
- In case of DCT - rndv completion is performed on one of the device dedicated DCIs (created by device).

Note that rendezvous offload is performed only for send_only/send_only with immediate / send_only with invalidate opcodes.

The below figures illustrates QPs which were connected to offloaded XRQ and passed to the device.

Figure 98: RNDV Completion - RC/XRC

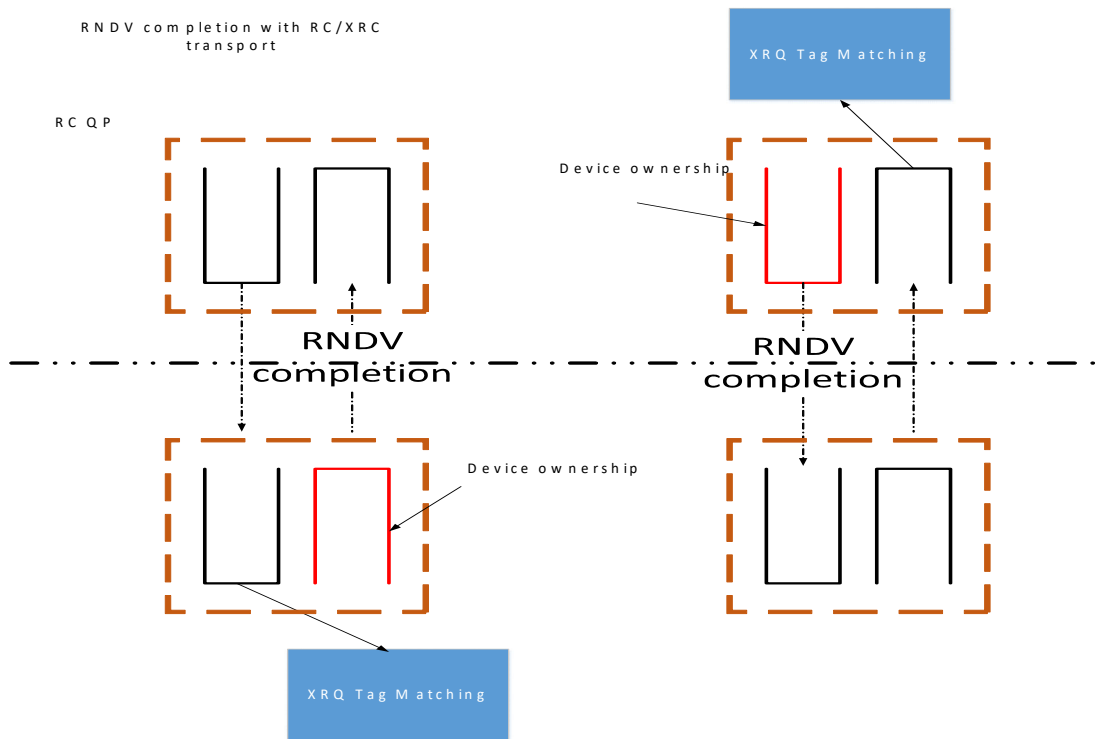


Figure 99: RNDV Completion - DC

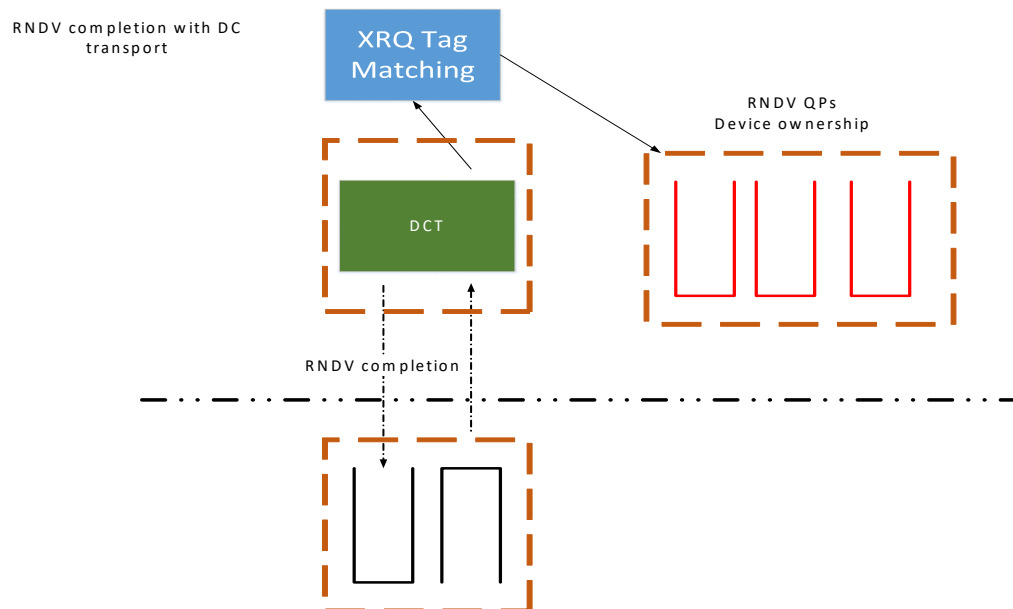
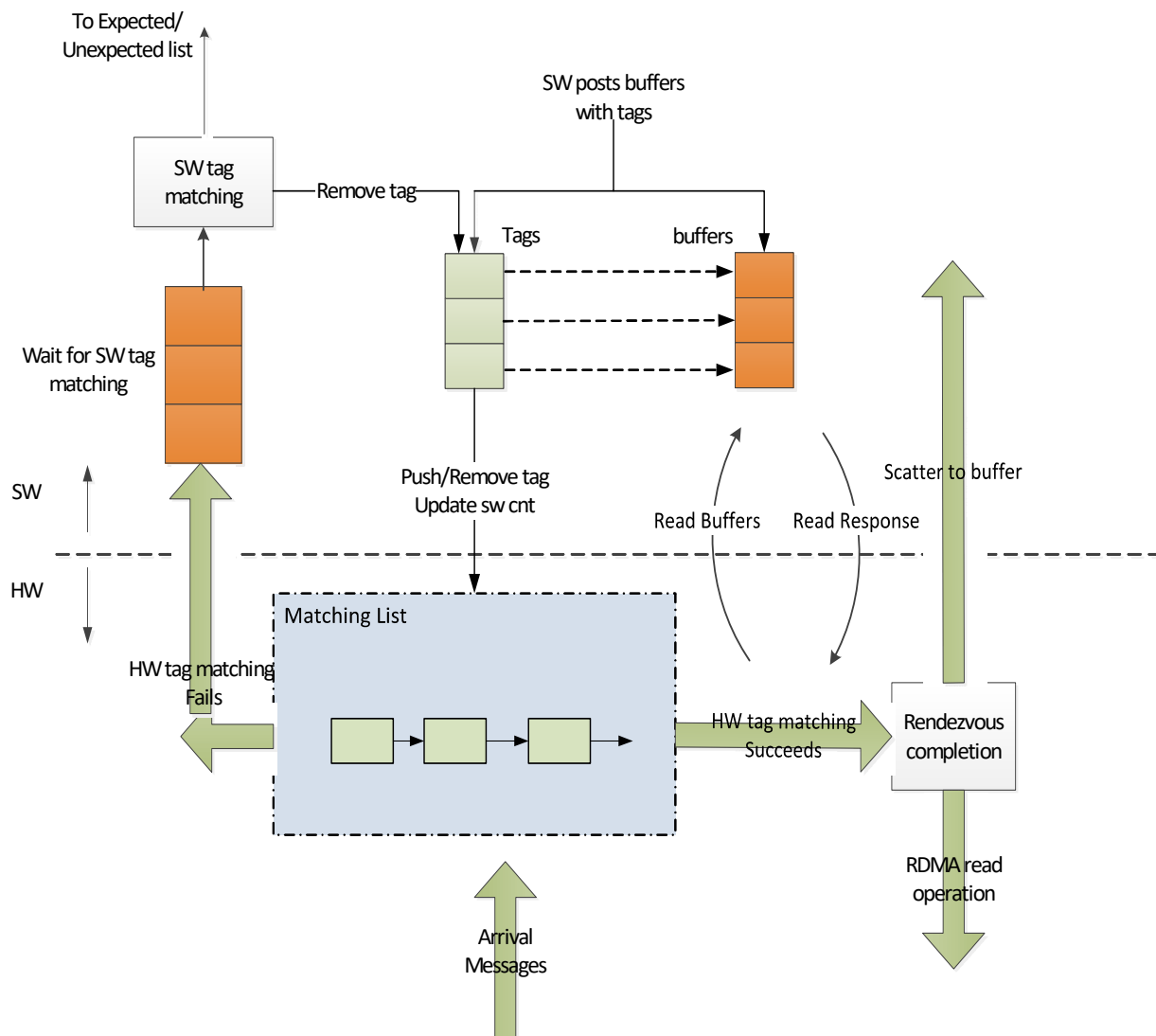


Figure 100, “High Level Arch of Tag Matching,” on page 990 describes the high level architecture of tag matching offload.

Figure 100: High Level Arch of Tag Matching



20.3.2 List Operations

SW can manage the matching list of the XRQ topology using list operation WQEs.

In order to execute each operation, SW writes a corresponding WQE to the QP send side (See [Section 8.9, “Work Queues,” on page 298](#)). Each operation will be directed to the XRQ which is connected to the QP receive side.

SW can request responder CQEs for list operation by setting the *xrq_cqe_req* in tag matching segment. List operation CQE arrives on XRQ CQ. Those CQEs do not consume WQE from UWQ. Therefore, in order to avoid CQ overrun, SW must ensure XRQ CQ has a place for the CQE (can be done by allocating larger XRQ CQ than transport needs and the extra space is the number of in flight list operation CQEs).

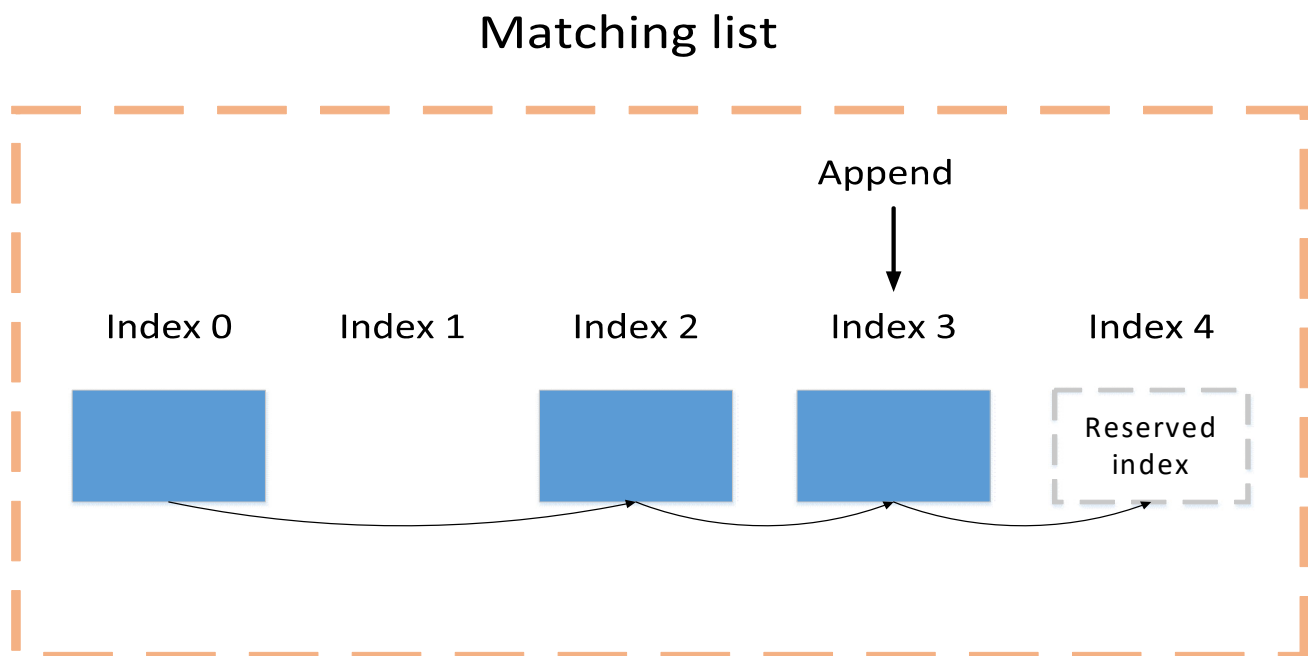
20.3.2.1 Append

Append inserts a new matching entry to the end of the matching list. In append, SW specifies the matching entry buffer using single scatter entry (see WQE format). In case SW wants to give buffer with higher number of scatter entries, it should perform UMR.

SW has full control over the matching list build, as it chooses to which index to insert the new matching entry. SW can choose the index to insert from the range $[0, \text{XRQ.matching\_list\_sz}-1]$. Index can be reused only when tagged message handling is over, i.e. SW received both “tag consumed” and “message arrived” CQEs.

For performance reasons, SW always specifies the next index it reserves for the next append operation and not the current index. Once SW has committed to next index, it cannot change it. In [Figure 101, “Matching List - Append Operation,” on page 991](#), SW reserves index 4 for the next append operation.

Figure 101: Matching List - Append Operation



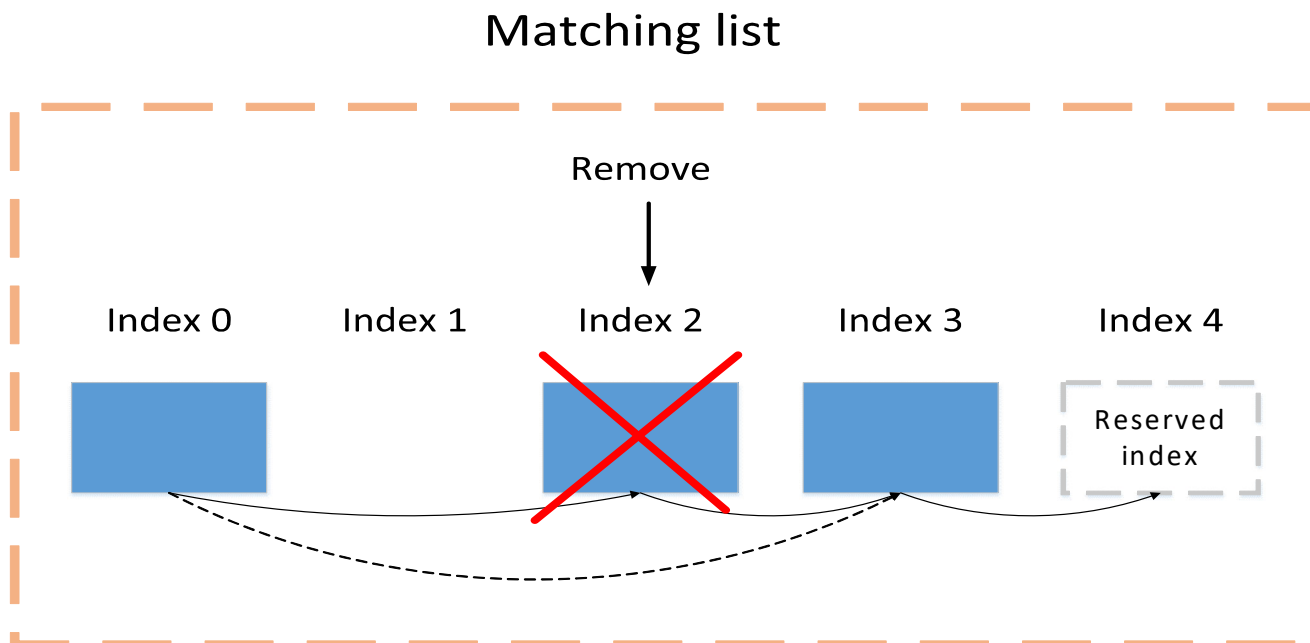
20.3.2.2 Remove

Remove operation removes a matching entry in a given index from XRQ matching list.

Remove operation can fail if the given index is empty. Index can be empty when a message consumed the matching entry before the remove operation arrived to device or index was never been used.

Figure 102, “Matching List - Remove Operation,” on page 992 describes a remove operation from index 2.

Figure 102: Matching List - Remove Operation



20.3.2.3 No operation

NOP passes through XRQ without changing its list nor state

20.3.3 Tag Matching CQE Generation

The device reports to SW about the completion of operations by CQEs. All tag matching CQEs are reported on TMC CQ. When QP is connected to XRQ, the QPC.cqn_rcv should be reserved.

The following CQEs are generated:

- **tag_consumed** - generated when matching entry was consumed from XRQ matching list. Upon receive, SW should remove the corresponding matching entry from its local list. Tag consumed CQE contains the message TMH. The device can merge both **tag_consumed** and **message_arrive** to a single CQE.
- **message arrive** - generated when data is valid in WQE from UWQ or in matching entry buffer. Message arrive CQE indicates the matching decision: expected, unexpected or no_tag (i.e. message is not carrying tag. e.g. **FIN message**). SW should update its local **sw_phase_cnt** only after handling **message_arrive_unexpected**.

- Message completion canceled – generated when the device has successfully received the message but cannot complete the offload. For example, when the device received rndv message but the matching entry size is too small for the fetched data. When receiving this CQE, the entire received message will be scattered to the matching entry.
- List operation CQEs - this type of CQEs contain the append, remove and NOP CQEs. These CQEs are generated when an operation on the matching list has been completed. They carry success/fail indication (for remove only) and the HW phase counter at the time the operation was executed.

Table 792 - Tag Matching CQE Summarize

Scenario	CQE Generated	Comment
Expected message arrived	Tag consumed Message_arrived_expected	The two may be merged into a single CQE
Unexpected message arrived	Message_arrived_unexpected	
Message that has no tag arrived	Message_arrived_no_tag	
Offloaded operation arrived which cannot be completed	Tag consumed Message_canceled	
List operation has been performed	Append/remove/nop	
Invalid offloaded operation arrived	Tag consumed sw rndv Message arrived expected	The two may be merged into a single CQE

20.4 Device Emulation

Note: This section is preliminary and subject to change.

Device Emulation allows for presenting a PCI device function that is not the standard Mellanox RDMA NIC/HCA. Such an emulated device function relies on software to perform its logic, beside some offloads to allow for an efficient data path.

The primary use case is for a SmartNIC, in which the emulation software can run in the embedded CPU while presenting a fully capable storage or network devices to the external host.

The PRM defines how to configure the device to present such emulated devices to the host, and how the emulation software should interact with the device in order to manage those emulated devices.

The emulated device exposed to the host can statically exist on boot by configuring the relevant NV_CONFIG fields (See Table 568, Table 580 and Table 576) or dynamically hotplugged/hotunplugged by creating or destroying a general object of type Device.

20.4.1 Initialization Flow

Below is the flow the emulation software on the Embedded CPU must follow in order to initiate and enable working with an emulated device.

- Check HCA_CAP.device_emulation_manager_#device_type (NVME/Virtio_net/virtio_blk...) according to the device type that is emulated. If set, the device will have emulation functions and SW will be able to manage them.
- For enhanced capabilities, SW should query Device Emulation Capabilities #device_type matching the corresponding device.
- Perform QUERY_EMULATED_FUNCTIONS_INFO command in order to get details about the already existing emulated functions, especially their vhca_ids. Use opcode to specify the device type of interest.
- **[Optional]** Check HCA_CAP.hotplug_manager to detect if vHCA can hotplug devices toward host. For enhanced hot plug capabilities, query hotplug manager capabilities.
- **[Optional]** If supported, hotplug new devices by creating a general object of type DEVICE. As part of the input mailbox SW will provide the initial registers values to be set on the emulated device BAR. When successful, the object id returned will contain the vhca_id of the newly created device on its 16 lsb.
- Create #device_type_DEVICE_EMULATION_OBJECT object for each vhca_id according to the device type to be managed. vhca_id of each function is provided either by QUERY_EMULATED_FUNCTIONS_INFO command or in the DEVICE object id (when creating it, hotplug the device).

Only a single emulated device object can be created per emulated vhca_id.

- **[Optional]** In order to be notified of changes on emulated device objects, SW should register to Object Change Event (See Section 8.20.9, “Object Change Event,” on page 481). Alternatively, SW should periodically query the emulation object to detect changes on the device bar. Some emulated devices (e.g. Virtio) are required to be armed in order to generate such an event.
- Complete the device specific initialization according to the device type. Refer to the relevant sections.
 - For NVME – see TBD
 - For Virtio-net or virtio-blk, see Section 8.27.19, “VIRTIO NET DEVICE EMULATION Object” and Section 8.27.20, “VIRTIO BLK DEVICE EMULATION Object”.
 - For Virtio emulation offload, see Section 20.7, “VIRTIO Emulation Offload”.

20.4.2 Teardown Flow

As a general guideline, the teardown should be in the reverse order of the initialization flow.

To teardown the emulated device:

- Perform device specific teardown flow.
 - For NVME – see TBD
 - For Virtio-net or virtio-blk, see Section 8.27.19, “VIRTIO NET DEVICE EMULATION Object” and Section 8.27.20, “VIRTIO BLK DEVICE EMULATION Object”.
 - For Virtio emulation offload, see Section 20.7, “VIRTIO Emulation Offload”.
- Destroy the emulated device object. After a successful call, the emulated device will be removed from the host.
- In case device is created by hotplug, destroy the device object.

20.4.3 FLR Flow

Upon FLR request from the host, the device clears the emulation object enabled bit.

In such cases, emulation SW is required to reset the emulated object by destroying and re-creating it.

During the time between the FLR to the emulation object reset, device will block any DMA operation from the host. MMIO to emulated device BAR will be stored by the device and reported to emulation SW upon query after the emulated object is restarted.

20.4.4 NVMe Emulation Offload

NVMe device emulation is one the supported device emulation. The primary use case is for a SmartNIC, in which the emulation software can run in the embedded CPU while presenting a fully capable NVMe device to the external host. It is up to the emulation software to decide how to handle the different Admin commands and IO requests. The PRM defines how to configure the

device to present emulated NVMe devices to the host, and how the emulation software should interact with the device in order to manage those emulated NVMe devices

20.4.4.1 Nvme Initialization Flow

This is the NVME specific initialization flow that the emulation software must follow in order to initiate and enable working with an emulated NVME device. SW must first follow the general emulation initialization flow described in Section 20.4.1, “Initialization Flow,” on page 994):

- For BlueField-1, create a `VHCA\_TUNNEL` for each NVMe `vhca_id` provided by `QUERY_EMULATED_FUNCTIONS_INFO` command. See Section 29.10.19, “QUERY_EMULATED_FUNCTIONS_INFO - Query Emulated Functions Information,” on page 1940
 - Only a single `VHCA\_TUNNEL` can be created per NVMe `vhca_id`
 - Execute `ENABLE_HCA` and `INIT_HCA` on the emulated function using the `VHCA\_TUNNEL`
 - Build the base steering tables as described in Section 8.18, “Flow Table,” on page 414.
- When `EMULATED_DEVICE.enable == 0`, SW should perform the TeardDown Flow. See Section 20.4.4.3, “TearDown Flow,” on page 997.
- Create the Admin Completion Queue by creating an `NVME_CQ` object with the relevant info from the registers.
 - Follow the `NVME_CQ` object description for the required flow.
- Create the Admin Submission Queue by creating an `NVME_SQ` object with the relevant info from the registers
 - Use SQE mode for the Admin queue
 - Use the `NVME_CQ` created above as NVMe CQ for this `NVME_SQ`
 - Follow the `NVME_SQ` object description for the required flow
- Create 2 QPs connected to each other. Assign one QP to the `NVME_SQ`. The second QP should stay in the hands of the Emulation Manager Software. This QP will be used to get Admin commands from the host, DMA the SQEs and data, and generate completions.
 - For BlueField-1, create the QP that is assigned to the `NVME_SQ` on the NVME function, using `VHCA_TUNNEL`. Then set Flow Table rules for that QP as described in the steering section.
- Raise `CSTS.RDY` register bit to send a signal to the host that the NVMe device is ready
- Process received Admin commands as needed

- Upon “Create IO CQ” and “Create IO SQ”, Admin commands create additional NVME_CQ and NVME_SQ objects respectively.
- Use Command Capsule mode for the IO queues
- For each NVME_SQ, create two QPs, one to be assigned to the NVME_SQ and one to stay in the hands of the Emulation Software
 - For BlueField-1, the QP assigned to the NVME_SQ should be created using the VHCA_TUNNEL, and proper steering rules should be set according to the steering section.
- Alternatively, create only one QP that is connected to the network, and assign it to the NVME_SQ. This is the “full offload” capability.
 - For BlueField-1, this QP should be created using the VHCA_TUNNEL, and proper steering rules may be needed according to Section 8.18, “Flow Table,” on page 414.
 - When connecting to external NVMe-oF target, the target side expects an NVMe-oF “CONNECT” message as the first message on that QP. This message and the corresponding response should be handled before assigning the QP to the NVME_SQ. For BlueField-1, assign the QP to the NVME_SQ first, then use the SEND_NVMMF_CC/QUERY_NVMMF_CC_RESPONSE to process that CONNECT message.
- Handle IO requests (if not working in “full offload” mode)

In Ethernet mode, the device that manages the emulated functions should be an SF that is opened to manage NVMe functions. This SF should be correctly configured, added to the e-Switch and allowing network access as needed.

20.4.4.2 Flow Table configuration

This section is only relevant for BlueField-1 with Ethernet, when using the “full offload” mode in which the QP assigned to an NVME_SQ is connected directly with a target in the network.

The QPs assigned to the NVME_SQ are created using a VHCA_TUNNEL, which means that they belong to the NVMe function. The requirement is for the traffic generated by the NVMe functions to seem like it is going out of the SF on the ECPF. This means that the traffic will have the network properties that are configured into that SF. Traffic coming from the wire will have destination properties of this SF as well. Therefore, special steering needs to be set up to divert packets generated in the context of the NVMe function (where the QP really is) to the SF, and back.

On the SF side, DV and DevX are used to play nicely with other possible applications on the SF. On NVMe function side, VHCA_TUNNEL is used to run PRM level commands on behalf of the NVMe emulated function.

Steps to set up steering - TBD.

20.4.4.3 TearDown Flow

To disable/tear down the emulation offload, SW should:

- Destroy all NVME_SQ objects. For those NVME_SQ objects that have QPs assigned to them, they are now back to software ownership, and can be destroyed as usual.
- Destroy the MKEYs that were given to this NVME_SQ.
- Destroy all NVME_CQ objects.
- Destroy VHCA_TUNNEL Object (if created for BlueField-1 only)
- Perform General Emulation Teardown Flow See Section 20.4.2, “Teardown Flow,” on page 995.

20.5 NVMe Express over Fabric (NVMe-oF) Target Application Offload

Note: This section is preliminary and subject to change. This feature is supported when `HCA_CAP.nvmf_target_offload==1`.

[NVMe is a protocol for communicating block storage IO requests over RDMA \(TBD pointer to spec?\).](#)

The NVMe-oF target offload described here provides the IO data path functionality of an NVMe over Fabrics Front-End Subsystem transferring the IO operations to NVMe PCIe Subsystems. In order to operate the offload, the software developer should open an XRQ context with offload type NVMe-oF. In typical use, XRQ will represent a front-end subsystem or a list of controllers within a subsystem that are aligned with respect to the attributes of the XRQ. The XRQ includes parameters such as queue depth, memory page size, in capsule data offset and so on.

Along with the XRQ, the user should provide staging buffer for the offload to stage the data on the way to the NVMe backend controller. This buffer should be accessible to the NVMe subsystems and the HCA. [\(TBD add pointer to XRQ PARAMs\)](#)

When creating XRQ with offload type NVMe, SW must pass in the XRQ WQ the physical addresses of these parts in the below order:

- Physical addresses of the RQ for receiving non-offloaded NVMe-oF commands
- Physical addresses of the staging buffer

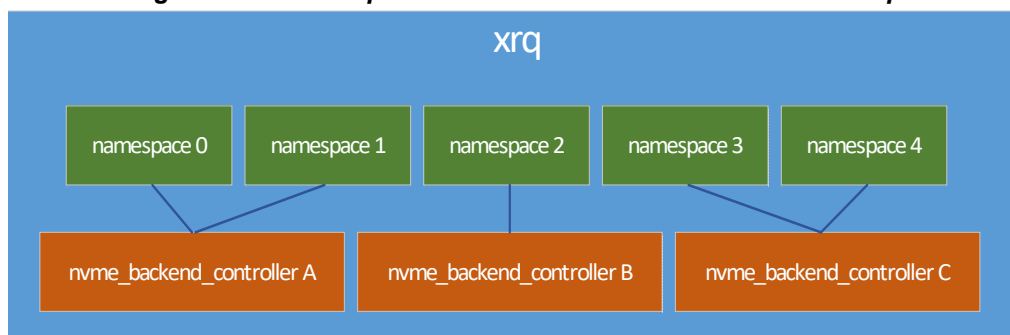
Note: The IOMMU addressing of the staging buffer should be identical between the HCA and the NVMe device.

In order to enable the XRQ to access NVMe backend subsystems, the user should associate at least a single *nvmf_backend_controller* with the XRQ. This context includes the required fields to enable accessibility to the NVMe backend subsystem. Examples for such fields would be the location of the NVMe submission and completion queues, location of the doorbell registers and so on. [\(TBD add pointer to backend ARAMs\)](#)

In order to enable the XRQ to expose several backend namespaces, the user should attach at least a single *nvme_namespace* to the XRQ and associate this namespace with one of the existing *nvmf_backend_controller* within this XRQ. The *nvme_namespace* includes fields which are specific per namespace and require enabling the abstraction of namespace within an NVMe subsystem. An example for such fields would be namespace block size, backend namespace identifier for abstracting more than one drive and so on. [\(TBD add pointer to namespace PARAMs\)](#)

The figure below shows the architecture of the offload, where the XRQ has the ability to expose multiple namespaces, each one of them associated with an *nvmf_backend_controller* (many to one).

Figure 103: Namespace and Backend Controller Relationship



The offloaded entities above are under the exclusive ownership of the device.

After the creation of the XRQ, in order to utilize the offload with a specific channel, the QP that is created for NVMe-oF communication should be associated upon creation with an XRQ that is configured for NVMe-oF target offload. This will direct command capsules to the offloaded enabled XRQ.

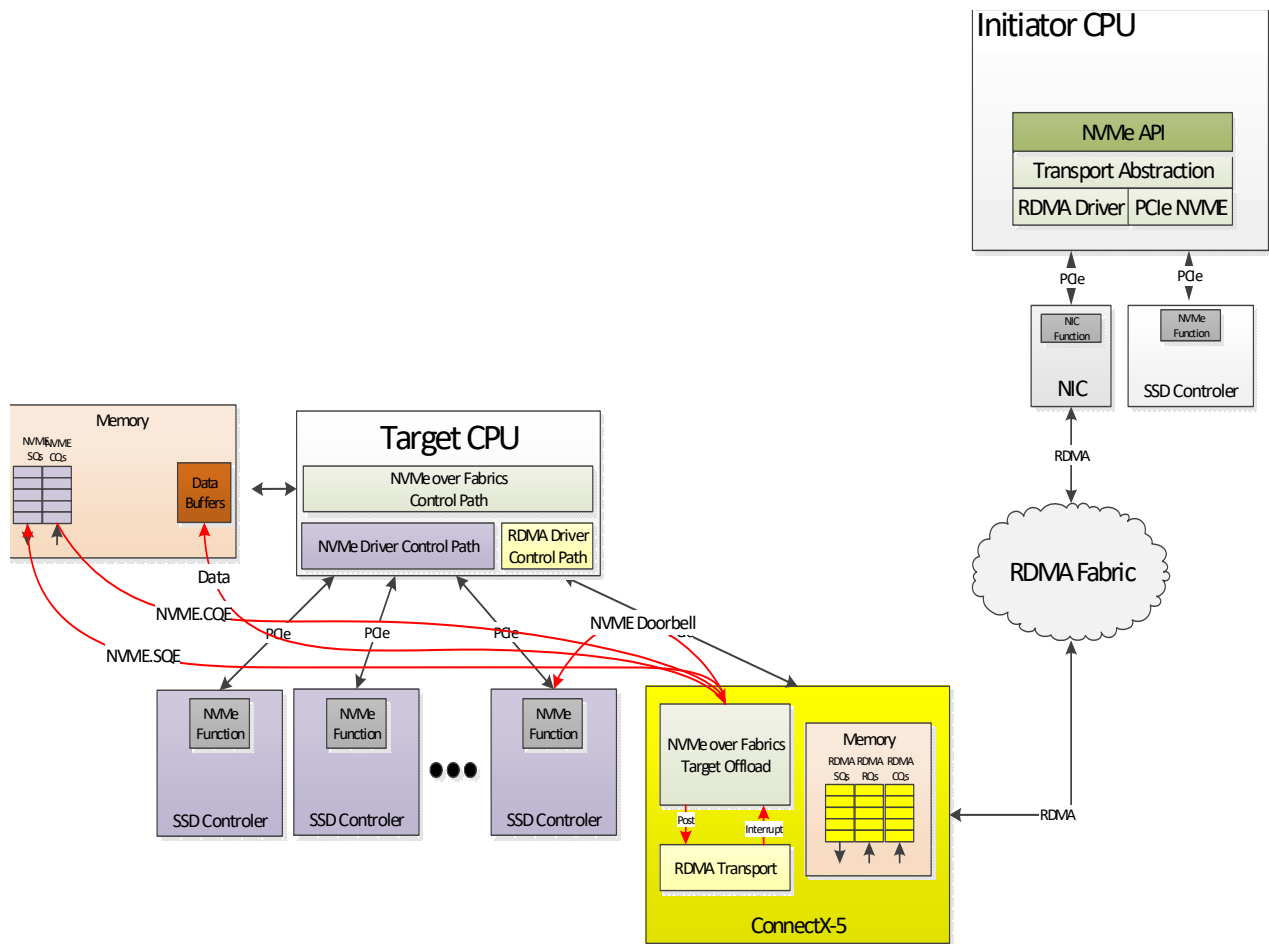
Offloaded NVMe CC will be processed by the offload engine and will be terminated with the backend controller associated with the namespace within the CC. Non-offloaded operations will be reported to the XRQ as usual.

Unsupported commands will be posted to the XRQ for further software processing, and CQEs will be reported to the XRQ CQ.

The operational flow of the offload works as follows:

- Upon receiving a command capsule (CC) to a QP associated with an XRQ configured for NVMe-oF target offload, the NVMe-oF offload engine is responsible for execution and completion of the command towards the NVMe-oF backend controller associated with the namespace of the command.
- For IO WRITE, data will be fetched from the client with RDMA READ. For IO READ, data will be placed in client memory with RDMA WRITE. For in-capsule IO WRITE, the data will be located in capsule. Intermediate buffering will be done using the staging buffer.
- Upon completion from the NVMe-oF backend controller, the offload sends the response capsule (RC) to notify the initiator that the transaction is done.

Figure 104: NVMe-oF Overview



20.5.1 NVMe-OF Report Back

This feature is intended to allow NVMf initiators to get reports on the pending IOs in the NVMf target (operated by Mellanox NVMf target offload feature). The reports should be part of each NVMf completion, and should include global information on all configured XRQs in the system.

The information should include the total known data that is in-flight per XRQ, and the total known pending requests in each XRQ. This is per function (which means per port).

It is understood that to avoid locks, the target implementation does not guarantee exact numbers.

In this design it is assumed that there is a configuration knowledge transferred between the target and initiators, such that the initiators are aware that the incoming NVMf completion is expected to be bigger, and to contain statistics on the internal resources of that target system, as defined here. The initiators are also aware of the different XRQs and how they are laid in the statistics buffer.

20.5.1.1 Increasing NVMF Completion Capsule

The NVMe-oF standard defines a field in the Identify Controller admin command response called "I/O Queue Response Capsule Supported Size (IORCSZ)" (32 bit):

This field defines the maximum I/O response capsule size in 16 byte units. The minimum value that shall be indicated is 1 corresponding to 16 bytes.

Apparently this is a non-negotiable field reported by the target.

From checking the existent NVMf host driver in Linux, the value of this field is ignored, as it is defined to be minimum 1 (16B).

Flow:

- Admin to enable "report back" feature on target side
 - Configure requested size (power of 2, most likely 64B or 128B)
 - Target will report IORCSZ of 64B or 128B (the values 4 or 8), or more
 - Target will populate first 16B with the completion by the spec, and the rest with the "report back" statistics
- Admin to enable "report back" feature on the initiators side
 - Initiator stack should allocate response capsules of the size published by the target IORCSZ. Otherwise, completions will be dropped as recv WQEs are set to 16B

20.5.1.2 Report Back Layout

The "report back" area consists of a 1B per XRQ created in the target system.

A configuration in the specific XRQ determines how the 8 bits are used between the two reported values.

- `XRQ.num_bits_for_in_flight_data-1 :0` - In flight data
- `7: XRQ.num_bits_for_in_flight_data` - Pending requests

Notes:

- If `num_bits_for_in_flight_data` is 0, in flight data is not reported, and all 8 bits are used for pending requests. If it is set to 8, pending requests is not reported and all 8 bits are used for in flight data.
- The place for each XRQ report within the report back space is configured in the XRQ.

If the highest place in the report back space is less than the available room in the response capsule, the rest of the capsule won't be sent (actual length will be shorter than the allowed IORCSZ).

20.6 Disk Encryption

The NIC can provide disk encryption services, allowing inline data encryption and decryption towards a remote or a local disk/NVDIM. With these abilities, the plugged out storage device's contents are unreadable without the encryption keys.

This support can be used to perform all disk encryption related flows in the NIC instead of using self-encrypting drives (SEDs). Having all encryption/decryption operations done in a single device can reduce cost and overheads of the related FIPS certification, as ConnectX-6 is FIPS 140-2 level 2 ready.

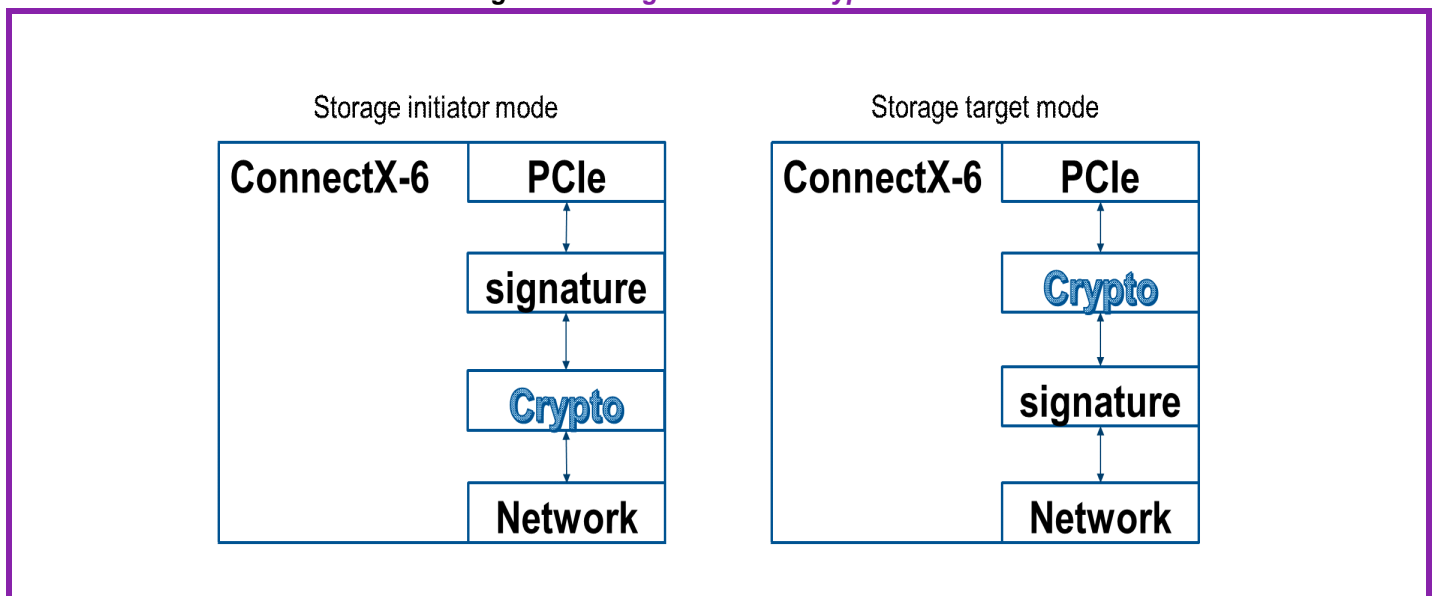
The inline encryption cipher is XTS-AES of 256/512 key size. The disk encryption operations can be performed inline along with additional storage-related offloads, such as signature, NVMeoF, etc.

The encryption operations are performed as transactions with single memory key representations. A transaction can include multiple RDMA Read/Write or Send requests. The minimal transaction size is the storage block size (disk block size). Data should be accessed contiguously within a memory key.

The inline encryption does not require network traffic to be aligned with the AES block size (128b).

To support both storage initiator and storage target modes, the ordering of the cryptographic and the data integrity operations is configurable, allowing the encryption to be performed closer to the target disk devices in both modes.

Figure 105: Signature and Crypto



20.6.1 Crypto Concepts

20.6.1.1 Import Methods

A device comes out of the Mellanox factory with a pre-defined import method. The import method defines how sensitive data (keys, credentials) should be loaded into the device, and may impose some extra production and runtime steps. For example, a device defined to work in Plaintext import method expects keys to be loaded in plaintext, and is ready for key management operations immediately.

A device with crypto engines defined to work in Wrapped import method expects keys for these engines to be loaded encrypted, and require a one-time (per device) manufacturing step to put those engines in Crypto Operational mode. It also requires authentication by logging in with credentials prior to software management of those keys.

20.6.1.2 Import Keys Encrypting Keys (Import_KEK)

When the device is defined to work in Wrapped import method, import_KEKs are used to wrap (encrypt) critical security parameters such as other keys and credentials, when those need to be passed between the device and the software.

The first import_KEK is provided once in clear text during the initial commissioning to operational mode changes. The other import_KEKs (and DEKs and credentials) are provided wrapped by an existing import_KEK. Once keys (or credentials) are passed to the device using an import_KEK, the key (or credential) will not maintain a relation to the import_KEK that was used to pass it. This implies that the initial import_KEK (provided in clear text during commissioning to operational mode transition) has no special meaning and can be later replaced by another import_KEK and destroyed if needed. import_KEKs may be stored persistently and securely on the device flash. Normally, a host will have a single operational import_KEK.

20.6.1.3 Data Encryption Keys (DEKs)

Data encryption keys (DEKs) are the keys used for data encryption/decryption operations.

DEKs reside in the ICM, which means that they disappear and require to be reloaded by the host software after every device boot.

20.6.1.4 Credentials

When crypto engines are defined to work in Wrapped import method, credentials are used to restrict cryptographic control operations to two roles: Officer (mandatory) and User (optional).

Credentials are delivered wrapped to the device (except when changing to operational mode), and are stored persistently on the device flash.

Every encryption-related control operation requires a credential to be provided, with the credential type (officer or user) corresponding to the operation executed, as described in the following table.

Table 793 - Credentials Permissions Summary

Operation	Officer	User
Get Crypto capabilities (No authentication is required)	V	V
Crypto Login/Logout	V	V
Add/Destroy/Query import_KEK	V	
Add/Destroy/Query Credential	V	
Add/Destroy/Query DEK	V	V

20.6.1.5 Commissioning and Operational Modes

When crypto engines are defined to work in Wrapped import method, they come out of the factory in Commissioning mode, thus cannot be used for crypto operations yet.

A dedicated tool is used for changing the mode from Commissioning to Operational while setting the first import_KEK and officer credential in plaintext. The mode is saved in flash so it is persistent between power cycles and resets. The current mode is visible in the crypto capabilities. Mode change may not immediately take effect upon the successful completion of the tool. Therefore, Admin should inspect the wrapped_crypto_operational bit to make sure it has changed.

20.6.1.6 MKeys

Memory Keys are used to describe a region in memory that can be later used in any RDMA transaction (SEND/RECV/READ/WRITE).

To encrypt/decrypt, a memory key can be assigned with crypto properties (when creating the key or later when using a UMR command). The most important property is a DEK index. When done, every access to the memory region by an RDMA transaction will result in on-the-fly encryption or decryption of the data.

20.6.1.7 KeyTags

A keytag is a piece of data encrypted together with a DEK.

When a DEK is referenced to by an MKEY.bsf through its index, the keytag is also supplied in the BSF as plaintext.

The hardware will decrypt the DEK (and the attached keytag) and will fail the operation in case the keytags do not match. This provides assurance that the key being used based on its index is the expected key. Software can choose not to use this feature by always supplying a keytag of 0.

20.6.1.8 Opaque

DEKs have an opaque property that can be set by the user to any value. It is not encrypted, however, it is signed together with the DEK and authenticated upon use of the DEK.

20.6.2 Crypto Workflow

20.6.2.1 Crypto Capabilities

- Check that `HCA_CAP.aes_xts==1` or other crypto engines capability bits in Crypto Capabilities is supported. See Table 1625, “Crypto Capabilities Layout,” on page 1802. If not set, the device does not have the respective crypto support. Otherwise, it might be that one of the self tests required for this engine has failed,
- Check the `failed_selftests` field in Crypto Capabilities is supported.
- Check `crypto_operational` capability bit in Crypto Capabilities. If not set, then the device import method requires a one time setting of the device to Crypto Operational mode, which was not yet done. Crypto cannot be used until then. Use the tool described in Section 20.6.2.2, “Move to Operational Mode,” on page 979 below.

20.6.2.2 Switching to Operational Mode

The device import method may require setting it to Crypto Operational mode before starting to use the crypto capabilities. For example: the “All engines wrapped” import method. Such a device comes out of Mellanox factory in “Commissioning” mode. In this mode, it is impossible to encrypt/decrypt or manage keys and credentials. Therefore, the first step should be switching it to Operational mode as follows.

Follow these steps:

1. Set `CRYPTO_OPERATIONAL` register to set the device in crypto operational mode (see Table 943, “`CRYPTO_OPERATIONAL` Register Layout,” on page 1135).
The input to this tool is:
 - The first Credential in plaintext
 - The first `import_KEK` in plaintext
2. Make sure command was executed successfully.
3. Query `CRYPTO_OPERATIONAL` register to make sure the device is in Operational mode.

From this point onwards, every command with sensitive data has to be encrypted.

In cases of power loss or reset during this process, the device may come back up with the mode unchanged. Host will need to repeat the above flow.

4. Start a crypto session (if needed)
5. Load the first DEK before encrypting/decrypting.

20.6.2.3 Starting a Crypto session

The device import method may require setting it to Crypto Operational mode before starting to use the crypto capabilities. For example: the "All engines wrapped" import method.

Create a [Crypto Login Object] to open such session before continuing to load DEKs, import_KEKs or credentials - see Table 302, "CRYPTO_LOGIN Object Format," on page 570.

The input to this object is:

- An existing crypto credential, either a User or an Officer role
 - The first officer credential was created in the previous section using the [Crypto To Operational mode] register.
 - Officer role is needed to manage import_KEKs and credentials. User role is sufficient for manipulating DEKs.
- An existing import_KEK pointer that will be used for wrapping all keys during this session
- The requested wrapping method for this session

It is possible to log in again with a different crypto role, import_KEK or wrapping method. It is also possible to log out of a crypto session.

20.6.2.4 Loading a DEK

At least one DEK is required in order to start actual encryption/decryption transactions.

1. Prepare a DEK.
 - Depending on the wrapped_import_method capability of the required engine, and options set in the DEK creation command, the DEK needs to be correctly constructed and may need to be wrapped using the import_KEK and wrapping method set with the [Crypto Login Object].
2. Create a [DEK object] to load the DEK to the device.

If executed successfully, A DEK would be available for actual encrypt/decrypt operations.

20.6.2.5 Running Data Crypto Operations

Encryption and decryption operations are done by attaching crypto properties to a memory region. Once done, every access to the memory via the crypto memory region will result in in-line encryption or decryption of the data.

1. Create Mkey with crypto_en==1 (usually done at initialization time, no crypto properties needed yet).
2. Post a UMR operation for the created MKEY with Crypto BSF to give the MR(memory region) crypto properties.
 - The crypto properties include the DEK to use and its keytag

3. Post IO operation with the MR - it can be any operation that has access to the MR: SEND/RECV/RDMAREAD/RDMA-WRITE.
4. Invalidate the MR as needed.

20.6.2.6 Updating Import_KEK

Occasionally, it may be required to update import_KEK. It may be due to organization policy in which keys are updated every fixed period of time, or it may happen as a result of a compromised key.

1. Create a new IMPORT_KEK object - see Table 297, “IMPORT_KEK Object Format,” on page 567.
 - Depending on the wrapped_import_method capability for the required engine, the new supplied import_KEK may need to be wrapped by the an existing import_KEK as set with the [Crypto Login Object]
2. If the old import_KEK that needs to be destroyed was used in the current [Crypto Login Object] (see Table 302, “CRYPTO_LOGIN Object Format,” on page 570), destroy the [Crypto Login Object] and re-create it with the new import_KEK.
3. Destroy the [IMPORT_KEK object](#IMPORT_KEK object) to delete the old import_KEK

20.6.2.7 Updating DEK

Occasionally, it may be required to update a DEK. It may be due to organization policy in which keys are updated every fixed period of time, or it may happen as a result of a compromised key.

The general process for DEK update is to create a new DEK, decrypt all data what was encrypted with the old DEK and re-encrypt it with the new DEK. Then the old DEK can be safely destroyed.

20.7 VIRTIO Emulation Offload

VIRTIO Emulation offload allows SW to offload the I/O operations of a VIRTIO Virtqueue, using the device, allowing an improved performance for its users. The list of supported VIRTIO devices types is reflected in general_obj_types in QUERY_HCA_CAP.

When supported by the device, SW can create a device specific VIRTIO_Q general object (for example: VIRTIO_NET_Q) while supplying all the relevant Virtqueue information (type, size, memory location, doorbell information, etc.). The device can then offload the I/O operation of this queue, according to its device type characteristics. For example, in VIRTIO_NET virtqueues, the device will offload the I/O operations by sending and receiving packets from its VIRTIO_NET Virtqueue, according to the VIRTIO Spec.

SW can query the VIRTIO Emulation Capabilities in QUERY_HCA_CAP to understand which features are supported. For example which VIRTIO Virtqueue layout type is supported (SPLIT or PACKED) and other Doorbell information.

VIRTIO_NET_Q object id can be used as a destination in TIR (inline_rqn) or in RQT (RQ List) (in addition to RQN) for steering purposes.

20.7.1 Initialization Flow for Offloading VIRTIO_NET Virtqueues

Below is the flow that software should follow in order to initialize and enable an offloaded VIRTIO_NET Virtqueues.

- Check HCA_CAP.general_obj_types.VIRTQ_NET_Q. If supported, execute QUERY_HCA_CAP with VIRTIO Emulation Capabilities op_mod
- When the VIRTIO_NET Virtqueues are ready in host and its driver is up and running, create a single Mkey that maps the memory address space of the Virtqueue application. When the Virtqueue resides in a VM, SW needs to create an MKEY that maps the Guest Physical address space (GPA). If needed, use indirect MKEYs, KSM and DUMP MKEYs utilities.
- For each VIRTIO_NET Virtqueue:
 - A single entry collapsed (cc) CQ with always-fired state and override ignore (oi) enabled
 - Create VIRTIO_NET_Q general object - See [Table 304, “VIRTIO_NET_Q Object Format,” on page 573](#)
 - SW may create the EQ/CQ relationship in any desired state and generate interrupt upon completion of I/O transactions for the VIRTIO_NET Virtqueue.
 - Query the umem_size field required by the emulation SW
 - Allocate/reserve a memory buffer as reported in umem_size
 - Modify the VIRTIO_NET_Q object and set state to RDY and provide the allocated buffer.
 - If send Virtqueue - Create TIS and configure TX steering as usual up to the TIS (without creating the SQs).
 - If Receive Virtqueue - create TIR/RQT and attach the VIRTIO_NET_Q objects Ids (instead of RQ number).

20.7.2 Teardown Flow

- Release all RX flow steering objects
- Destroy all VIRTIO_NET_Q
- Release/regain the UMEM allocated space

20.7.3 Live Migration Flow

- Create a single Mkey that maps the memory address space of the VHOST live migration log file.
- Modify VIRTIO_NET_Q object and provide vhost_log_page, dirty_bitmap_mkey, dirty_bitmap_size, dirty_bitmap_addr and dirty_bitmap_dump_enable.
- Modify VIRTIO_NET_Q object and move state to SUSPEND
- Query VIRTIO_NET_Q and get hw_available_idx and hw_used_idx.

20.7.4 Error Flows

In case of error the device will move the VIRTIO_NET_Q object to ERR and will fire an Affiliated Object Change Event.

Upon receiving this event for a specific VIRTIO_NET_Q SW should:

- QUERY the VIRTIO_NET_Q
- Query VIRTIO_NET_Q, verify state ==ERR and read the syndrome and check it meaning.

21 FPGA Device

21.1 Overview

The FPGA device consists of a HCA linked to a Xilinx FPGA via an Ethernet connection. The FPGA implements special packet processing logic that offloads CPU processing to the device. The FPGA has access to its own memory that is mainly used to maintain contexts. A special RC QP between the host and the FPGA is used for control traffic. There are 2 flashes connected to the FPGA, one stores the factory default and another one stores the sandbox. Each flash may store multiple versions of the same image.

21.2 Initialization and Tear Down Flow

After initialization, the FPGA device is loaded with last image that used. The sandbox is initialized in bypass state, and SW is responsible to turn off the bypass using `FPGA_CTRL` register. On teardown, SW should turn on the bypass again.

21.3 FPGA Port

All physical port operations managed via the regular network port registers (implicitly). (See Section 28.19, “Network Ports Registers,” on page 1140).

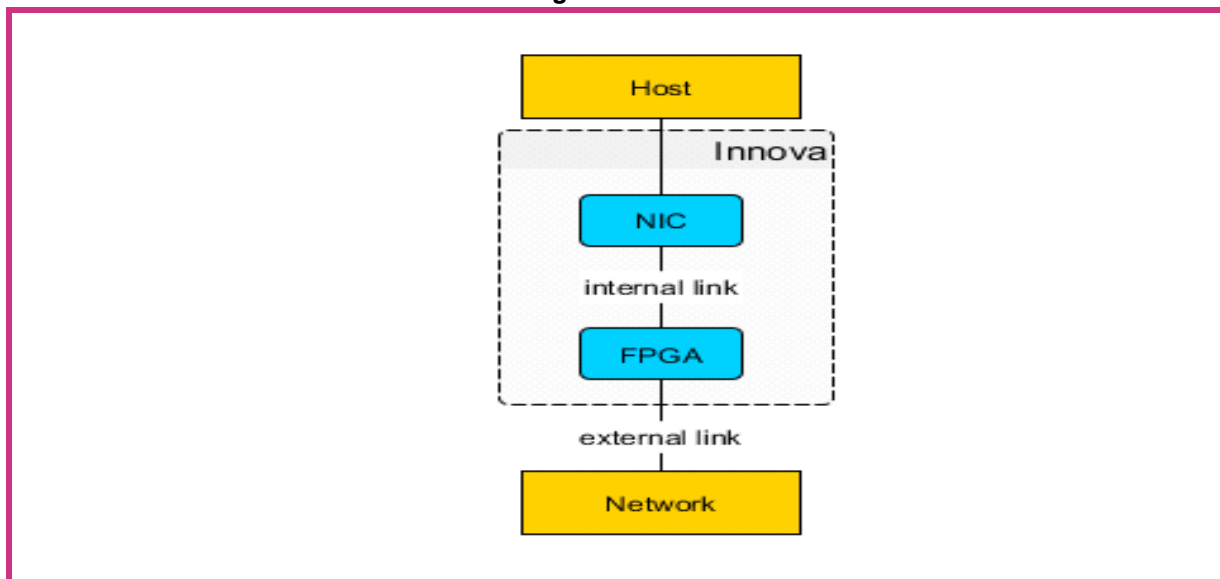
- On query operation, the device reports the shared capabilities between all of the 3 ports.
- On write/configuration operation, all of the 3 ports will be modified.

For example, if FPGA supports only 40Gbps on both ports while the HCA supports 10/40/56Gbps, then only 40Gbps will be reported to the user via a query operation.

There are two exceptions to the rule above:

- Port state configuration - The device will keep the internal link (FPGA-HCA) always up, and in case the internal link is down, a fatal FPGA event is raised (See Section 21.4.3, “FPGA Async Events,” on page 1013). Each operation on the physical port state will be performed on the external FPGA port.
- PPCNT - The default of PPCNT register will report the external network port (FPGA-Network). SW can still read the counters of the internal ports by setting *port_type* in PPCNT register.

Figure 106: FPGA Ports



21.4 Software Interface

21.4.1 FPGA QP

See Section 8.25, “FPGA QP,” on page 530.

21.4.2 FPGA Shell QP Packet Format

This section describes the packet format to communicate with the Shell QPs in order to read/write FPGA memory and registers space.

Table 794 - FPGA Shell QP Packet Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0		
version			syndrome						type									tid						00h								
len																																04h
address																																08h-0Ch
data[0]				data[1]				data[2]				data[3]				10h																
data[4]				data[5]				data[6]				data[7]				14h																
...																																::

Table 795 - FPGA Shell QP Packet Format Fields

Offset	Bits	Name	Description
00h	31:28	version	Version number
	27:24	syndrome	Syndrome. TBD
	19:16	type	Message Type. 0x0: DDR_Read 0x1: DDR_Write 0x2: DDR_Read_Response 0x3: DDR_Write_Response
	7:0	tid	Transaction id. Should be incremented for each transaction.
04h	31:0	len	Size of data field in granularity of bytes. Must be <= Shell_caps.max_fpga_qp_msg_size. See Table 893, “Shell_Caps Layout,” on page 1107.
08h-0Ch	64	address	Must be aligned to 4 bytes. Note that address+len combination must not cross 4KB boundary. i.e. the first byte and the last byte must be in same 4KB block.
10h-...	31:24	data[...]	Data in to write/read.

21.4.3 FPGA Async Events

See Section 8.20.15, “FPGA Events,” on page 504

21.5 FPGA Sandbox

TBD - Boris

21.5.1 Overview

TBD - Boris.

21.5.2 FPGA IPsec Sandbox

The FPGA IPsec sandbox implements symmetric cryptographic and authentication offload for IPsec traffic which frees CPU cycles.

21.5.2.1 IPsec Initialization

During initialization all SA entries are cleared. The memory and caches are reset. Finally, bypass is turned off and the device is ready for offload.

To initialize the IPsec sandbox the following commands must be issued using the FPGA_CTRL register:

- BypassSandboxOn
- Reset
- BypassSandboxOff

21.5.2.2 IPsec Teardown

During teardown bypass should be turned on such that no traffic will be offloaded.

To teardown the IPsec sandbox the following commands must be issued using the FPGA_CTRL register:

- BypassSandboxOn

21.5.2.3 IPsec Capabilities

The following capabilities are valid only when FPGA_CAP.ieee_vendor_id == Mellanox and FPGA_CAP.sandbox_product_id == IPsec (See Table 891, “FPGA_CAP Register Layout,” on page 1103).

Table 796 - IPsec_Basic_Capabilities Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 797 - IPsec_Basic_Capabilities Structure Field Descriptions

Offset	Bits	Name	Description	Access

Table 798 - IPsec_Extended_Capabilities Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
encapsulation																																00h	
																						ipv4_fragment	ipv6	ecn	lso	transport_and_tunnel_mode	tunnel_mode	transport_mode	ah_esp	esp	ah	ipv4_options	004h
																						auth_alg											08h
																						enc_alg											0Ch

Table 798 - IPsec_Extended_Capabilities Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sa_cap																															10h	
																number_of_ipsec_counters																14h
ipsec_counters_start_addr																															18h-1Ch	
																															20h-3Ch	

Table 799 - IPsec_Extended_Capabilities Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	encapsulation		
04h	10	ipv4_fragment	If set, IPv4 fragments are supported.	
	9	ipv6	If set, IPv6 offload is supported.	
	8	esn	If set, IPsec extended sequence numbers is supported.	
	7	lso	If set, Large Send Offload by FPGA is supported.	
	6	transport_and_tunnel_mode	If set, IPsec in tunnel and transport modes combination offloading is supported.	
	5	tunnel_mode	If set, IPsec in tunnel mode offloading is supported.	
	4	transport_mode	If set, IPsec in transport mode offloading is supported.	
	3	ah_esp	If set, AH and ESP combination offload is supported.	
	2	esp	If set, encapsulating security payload (ESP) offload is supported.	
	1	ah	If set, authentication header (AH) security offload is supported.	
	0	ipv4_options	If set, IPsec offload of packets with IPv4 options is supported.	
08h	31:0	auth_alg	Supported types of IPsec authentication algorithms (Bit-mask) Bit 0: AES_GCM_128 Bit 1: AES_GCM_256	
0Ch	31:0	enc_alg	Supported types of IPsec encryption algorithms (Bitmask) Bit 0: AES_GCM_128 Bit 1: AES_GCM_256	

Table 799 - IPsec_Extended_Capabilities Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	sa_cap	The number of SA bundles, which might include ESP or AH or both, that can be offloaded to the NIC.	
14h	15:0	number_of_ipsec_counters	Number of supported IPsec counters. This first counter allocated in the address which is reported by <i>ipsec_counters_start_addr</i> . The layout of the counters is reported in TBD	
18h-1Ch	64	ipsec_counters_start_addr	Address where first IPsec counter is allocated. The total number of counters reported by <i>number_of_ipsec_counters</i> .	

21.5.2.4 IPsec Counters

The table below describes the supported IPsec counters. The address where this table starts and the number of valid counters are reported by IPsec_Extended_Capabilities.ipsec_counters_start_addr. and IPsec_Extended_Capabilities.number_of_ipsec_counters.

TBD- add figure of counters.

Table 800 - IPsec_COUNTERS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dec_in_packets																																00h-04h
dec_out_packets																																08h-0Ch
dec_bypass_packets																																10h-14h
enc_in_packets																																18h-1Ch
enc_out_packets																																20h-24h
enc_bypass_packets																																28h-2Ch
drop_dec_packets																																30h-34h
failed_auth_dec_packets																																38h-3Ch

Table 800 - IPsec_COUNTERS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
drop_enc_packets																															40h-44h	
success_add_sa																															48h-4Ch	
fail_add_sa																															50h-54h	
success_delete_sa																															58h-5Ch	
fail_delete_sa																															60h-64h	
dropped_cmd																															68h-6Ch	

Table 801 - IPsec_COUNTERS Fields

Offset	Bits	Name	Description	Access
00h-04h	64	dec_in_packets	Total packets received for decryption by FPGA.	
08h-0Ch	64	dec_out_packets	Number of packets that were received for decryption, decrypted and successfully authenticated by FPGA.	
10h-14h	64	dec_bypass_packets	Number of packets that were bypassed by FPGA in decryption direction.	
18h-1Ch	64	enc_in_packets	Total packets received for encryption by FPGA.	
20h-24h	64	enc_out_packets	Number of packets that were received for encryption, encrypted and successfully authenticated by FPGA.	
28h-2Ch	64	enc_bypass_packets	Number of packets that were bypassed by FPGA in encryption direction.	
30h-34h	64	drop_dec_packets	Number of packets dropped by decryption engine. This can be as a result of having inband metadata in packet or corrupted decryption.	
38h-3Ch	64	failed_auth_dec_packets	Number of packets dropped by decryption engine due to authentication issue.	

Table 801 - IPsec_COUNTERS Fields

Offset	Bits	Name	Description	Access
40h-44h	64	drop_enc_packets	Number of packets dropped by encryption engine. This can be as a result of more VLAN tags than the number supported by FPGA, having inband metadata or miss in SADB.	
48h-4Ch	64	success_add_sa	Total amount of SAs successfully added by FPGA.	
50h-54h	64	fail_add_sa	Total amount of failed SA add commands by FPGA. This can be a result of adding an already valid SA.	
58h-5Ch	64	success_delete_sa	Total amount of SAs successfully removed by FPGA.	
60h-64h	64	fail_delete_sa	Total amount of failed SA remove commands by FPGA. This can be a result of remove command on invalid SA	
68h-6Ch	64	dropped_cmd	Total amount of failed command. This can be a result of failure to parse command.	

21.6 MMO - Memory to Memory Offloads

The device may support various Memory to Memory offloads. Each MMO type has its own HCA_CAP field/s as specified in the section of each offload. All MMOs are operated by posting WQEs of type MMO (Opcode=MMO), and the MMO type is indicated by the WQE's OPC_MOD field.

The MMO WQE has 4 segments:

- Control Segment - when opcode = MMO and OPC_MOD indicates the MMO type. Also, ctrl_general_id includes MSB of the **mmo_control**.

- MMO METADATA Segment - See [Section 8.9.4.1.19, “MMO Metadata Segment,”](#) on [page 346](#).
- Input buffer Data Segment - See [Table 80, “Memory Pointer Data Segment Format,”](#) on [page 340](#).
- Output buffer Data Segment - See [Table 80, “Memory Pointer Data Segment Format,”](#) on [page 340](#).

A device MMO WQE execution includes these steps:

- Read data from the input buffer.
- Optionally read data from the METADATA buffer indicates by MMO METADATA Segment.
- Optionally read `mmo_control. 32 MSB in CTRL_SEGMENT.ctrl_general_id` and 32 LSB in MMO METADATA Segment.
- Write data result to the output buffer.
- Optionally write data to the METADATA buffer indicated by MMO METADATA Segment.
- Generate a CQE.

21.6.1 REGEXP MMO

This feature is supported when `HCA_CAP.regexp==1`.

Before using the RegExp offload, SW must first configure its registers and DataBase using the `SET_REGEX_PARAMS` and `SET_REGEX_REGISTER` commands. See [Section 29.11, “REGEXP Commands,”](#) on [page 1942](#).

The RegExp offload logic is comprised of a number of engines as reported in `HCA_CAP.regexp_num_of_engines`. SW must configure all engines before starting to operate them.

Each engine has an internal configuration register space (CRSPACE) of size $2^{\text{HCA\_CAP.regexp\_log\_crspace\_size}}$. Each register is 32 bits wide. The `QUERY_REGEX` and `SET_REGEX` commands are used to access the engine registers. The content of these registers is implementation-specific, thus, opaque at PRM level.

The `SET_REGEX_PARAMS` command is used to configure additional RegExp engine parameters, such as the `umem_id` and `umem_offset` specifying the DMA address range of the RegExp rules database. The `QUERY_REGEX_PARAMS` may be used to query those engine parameters.

A RegExp WQE is an MMO WQE (opcode=MMO) with OPC_MOD=REGEXP.

The RegExp WQE mmo_control field content/format(TODO: add description for each field):

Table 802 - REGEXP MMO Control Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
		le																													00h
																															04h

Table 803 - REGEXP MMO Control Fields

Offset	Bits	Name	Description
00h	29	le	
	27:16	subset_id_0	
	11:0	subset_id_1	
04h	31:28	ctrl	
	27:16	subset_id_2	
	11:0	subset_id_3	

The input buffer of a RegExp WQE contains data where the RegExp offload engines should look for signatures.

The METADA buffer of a RegExp WQE is 64 bytes that is used for output only. The format of the offload-type-specific part of the METADATA buffer of a RegExp WQE: [\(TODO: add description for each field\):](#)

Table 804 - REGEXP MetaData Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
rof_version																latency_count																
instruction_count																primary_thread_count																
match_count								detected_match_count								status																
job_id																																
																																10h-1Ch

Table 805 - REGEXP MetaData Fields

Offset	Bits	Name	Description
00h	31:16	rof_version	
	15:0	latency_count	
04h	31:16	instruction_count	
	15:0	primary_thread_count	
08h	31:24	match_count	
	23:16	detected_match_count	
	15:0	status	
0Ch	31:0	job_id	

The output buffer of a RegExp WQE contains a list of suspected signatures: (TODO: tailor this description...).

21.7 IPsec Offload

21.7.1 Overview

Internet Protocol Security (IPsec) offload allows the NIC to accelerate IPsec AES-GCM 128/256 encryption/decryption (Crypto Offload) and to encapsulate/decapsulate data in ESP packets (Full Offload). IPsec offload handles packets inline - as they go through the NIC. If a packet cannot be handled by IPsec offload, then a software fallback should handle the packet. For example, IP fragments must be handled by a software fallback.

Two levels of offload are supported:

Full Offload

- ESP payload Packet Encryption/Decryption
- ESP payload authentication
- ESP Encapsulation/Decapsulation
- ESP sequence numbers and replay protection handling

For inbound packets, the host receives plaintext packets (for instance ETH|IP|TCP) while on the network these packets are encrypted and encapsulated within an ESP protocol (transport or tunnel). For outbound packets the host sends plaintext packets (for instance ETH|IP|TCP) while on the network these packet are encrypted and encapsulated with an ESP protocol (transport or tunnel). The hardware handles all the IPsec data-path.

Crypto Offload

- Packet Encryption/Decryption
- ESP payload authentication (without modifying the headers.)

The host always sends/receives formatted ESP packets with plaintext data instead of the cipher-text data, all other fields are unmodified.

21.7.2 Driver Initialization

At driver initialization, no contexts are created by default, without user interaction. For the initialization of the IPsec offload flows see the following sections. At initialization, driver should query whether IPsec offload is available through `QUERY_HCA_CAP`. If `HCA_CAP.IPsec_Crypto` is enabled then crypto offload is enabled. If `HCA_CAP.IPsec_Full` is enabled, then full offload is enabled. The driver should indicate to the host the supported offload through the network device capabilities. Otherwise, stop IPsec initialization.

Note: For E-Switch manager that performs IPsec offloads, these offloads must be disabled for the VFs using TBD, since packet may only be encrypted or decrypted by the device once.

21.7.3 Key management

Any key that is used with TLS offload must be created first by `CREATE_GENERAL_OBJ` command for Encryption key object (See Section 8.26.13, “Encryption Key Object,” on page 517). The Encryption key object handle is used later when creating the IPsec Offload object.

21.7.4 Crypto offload for transmitted packets

For Crypto offload on the transmit side, an IPsec context must be created using `CREATE_GENERAL_OBJ` (*full_offload* disabled). Then, to perform the offload on a flow of packets, a Flow Table Entry in “NIC transmit” Flow table should be added and point to the IPsec context. The driver is responsible to only send valid packets for Encryption offload: packets with ESP header, ESP trailer and plaintext data.

For example, driver can match Ethertype and IP protocol to verify the packet is an ESP packet, and optionally match ESP.SPI, Source IP and destination IP to define the specific flow. Another option is to classify packets for Encryption offload in SW and use the value in *flow_table_metadata* WQE field (loaded into metadata register type A) to indicate the classification result.

An example for a Flow Table configuration performing Encryption IPsec Crypto offload:

Table 1:

1. (Ethertype = IPv4) & (IP.protocol = ESP) & (ESP.SPI = 0x1234) --> actions: IPsec Offload, Allow

21.7.4.1 Trailer insertion

Crypto offload requires formatting the ESP packet by SW. The Trailer insertion can be offloaded to the device by the WQE (See Trailer Insertion), It is possible to send a fully formatted plaintext ESP packet without a trailer and let HW insert it.

The trailer insertion fields of the WQE should be set as follows:

1. Trailer Header association – set according to the header to which the trailer belongs –
 - 1 for normal ESP packet: ETH|IP|ESP,
 - 2 for ESP over UDP: ETH|IP|UDP|ESP
 - 3 for encapsulations of ESP packets such as: ETH|IP|UDP|VXLAN|ETH|IP|ESP
 - 4 for encapsulations of ESP over UDP such as: ETH|IP|UDP|VXLAN|ETH|IP|UDP|ESP
2. Trailer contents – the contents should contain all the ESP trailer as it is expected to appear on the wire – padding, pad length, next IP protocol, and an authentication tag place-holder, which will be updated by the Encryption operation.
3. [Internal] due to HW bug, Trailer alignment – set to no alignment (0)

21.7.4.2 Checksum Offload

TCP/UDP checksum offload for packets undergoing IPsec Encryption using Crypto Offload must use Software Parsing (SWP) to indicate the location of the relevant headers in the packet.

21.7.4.3 LSO Offload

LSO offload using The LSO_SEND WQE can be combined with IPsec Crypto offload. However, all TCP segments of the LSO must have the same length. In case the TCP payload is not a multiple of MSS, the payload should be split into multiple WQEs.

In addition, Trailer insertion offload and Software Parsing (SWP) must be used for LSO_SEND WQEs when Crypto offload is required.

21.7.4.4 LRO Offload

Decrypted IPsec packets, in Crypto offload mode are not be merged by LRO

21.7.4.5 TLS

When using TLS with IPsec Crypto offload, the trailer insertion offload must be used for all packets.

21.7.5 Crypto offload for received packets

For Crypto offload on the receive side, an IPsec context must be created using CREATE_GENERAL_OBJ (**full_offload** disabled). Then, to perform the offload on a flow of packets, a Flow Table Entry in “NIC receive” Flow table should be added and point to the IPsec context. The driver is responsible to only send valid packets for Decryption offload: packets with ESP header, ESP trailer and encrypted data.

For example, driver can match Ethertype and IP protocol to verify the packet is an ESP packet, and optionally match ESP.SPI, Source IP and destination IP to define the specific flow.

In addition to decrypting the packet, the IPSEC action will also fill the *ipsec_syndrome* field (see IPsec Syndromes) with a syndrome which can be matched by a following Flow Table Entry to validate the success of the decryption.

An example for a Flow Table configuration performing Decryption IPsec Crypto-offload:

Table 1:

1. (Ethertype = IPv4) & (IP.protocol = ESP) & (ESP.SPI = 0x1234) --> actions: IPsec Offload, goto table 2

Table 2:

1. (ipsec_syndrome = OK) --> actions: Reformat packet (DEL ESP), Forward to destination TIR (for decrypted packets)
2. * --> actions: Forward to Exception Flow TIR (for encrypted packets)

21.7.6 Full offload for transmitted packets

For Full offload transmit side, an IPsec context must be created using CREATE_GENERAL_OBJ (**full_offload** enabled). Then, to perform the offload on a flow of packets, a Flow Table Entry in a Flow Table (for example, the “E-Switch Forwarding Data Base”) should be added and point to the IPsec context. The steering entries that point to the IPsec action must verify that the packets have the IP header.

While Encrypting the packet, the IPSEC action will also fill the ASO return register field with a syndrome (see IPsec Syndromes) which can be matched by a following Flow Table Entry to validate the success of the Encryption.

An example for a Flow Table configuration performing Encryption IPsec Full offload:

Table 1:

1. Match 5-tuple --> actions: Reformat packet (ADD ESP), IPsec Offload, goto table 2

Table 2:

1. (ASO return value register = OK) --> actions: Forward to Uplink (Encrypted packet)
2. * --> actions: Count, Drop (failed encryption)

21.7.6.1 Packet Reformat

IPsec Full offload for encryption may use one of the following packet reformats:

- Add an ESP header and an ESP trailer for Transport mode:
 - original packet: MAC1|IP1|Data
 - reformatted packet: MAC1|IP1|ESP|Data|ESP-trailer
 - supported in Flow Table according to HCA_CAP.reformat_add_esp_trasport
- Remove L2 header and encapsulate the packet with a tunneling header (should include ESP header) and add an ESP trailer
 - original packet: MAC1|IP1|Data
 - reformatted packet: Encapsulation Header|IP1|Data|ESP-trailer
 - supported in Flow Table according to HCA_CAP.reformat_l2_to_l3_esp_tunnel
- Add an ESP header over UDP and an ESP trailer:
 - original packet: MAC1|IP1|Data
 - reformatted packet: MAC1|IP1|UDP|ESP|Data|ESP-trailer
 - supported in Flow Table according to HCA_CAP.reformat_add_esp_transport_over_udp

Note: The above reformat types should not be used for ESPoIPv6 packets with Extension headers.

21.7.7 Full offload for received packets

For Full offload on the receive side, an IPsec context must be created using CREATE_GENERAL_OBJ (**full_offload** enabled). Then, to perform the offload on a flow of packets, a Flow Table Entry in a Flow Table (for example, the “E-Switch Forwarding Data Base”) should be added and point to the IPsec context. The driver is responsible to only send valid packets for Decryption offload: packets with ESP header, ESP trailer and encrypted data.

For example, driver can match Ethertype and IP protocol to verify the packet is an ESP packet, and optionally match ESP.SPI, Source IP and destination IP to define the specific flow.

In addition to decrypting the packet, the IPSEC action will also fill the `ipsec_syndrome` and the ASO return register field (see IPsec Syndromes) with syndromes which can be matched by a following Flow Table Entry to validate the success of the decryption.

An example for a Flow Table configuration performing Decryption IPsec Full offload: Table 1:

1. (Ethertype = IPv4) & (IP.protocol = ESP) & (ESP.SPI = 0x1234) --> actions: IPsec Offload, goto table 2

Table 2:

1. (`ipsec_syndrome` = OK) & (ASO return value register = OK) --> actions: Reformat packet (DEL ESP), Forward to destination VPort (for decrypted packets)
2. * --> actions: Forward to E-Switch manager VPort (for encrypted packets fallback)

21.7.7.1 Packet Reformat

IPsec Full offload for decryption may use one of the following packet reformat:

- Remove the ESP header and the ESP trailer for Transport mode:
 original packet: MAC1|IP1|ESP|Data|ESP-trailer
 reformatted packet: MAC1|IP1|Data
 supported in Flow Table according to `HCA_CAP.reformat_del_esp_transport`
- Remove the ESP trailer, remove the tunneling header and add a new L2 header:
 original packet: Encapsulation Header|IP1|Data|ESP-trailer
 reformatted packet: MAC1|IP1|Data
 supported in Flow Table according to `HCA_CAP.reformat_l3_esp_tunnel_to_l2`
- Remove the ESP header, the UDP header and the ESP trailer:
 original packet: MAC1|IP1|UDP|ESP|Data|ESP-trailer
 reformatted packet: MAC1|IP1|Data
 supported in Flow Table according to `HCA_CAP.reformat_add_del_transport_over_udp`

Note: The above reformat types should not be used for ESPoIPv6 packets with Extension headers.

21.7.8 Extended Sequence Number (ESN)

ESN is defined in multiple RFCs, for example:

- RFC4106 for AES-GCM and ESN.
- RFC4301 for ESN and replay window interaction.

ESN influences both the AAD and the IV of packets. The packet's AAD consists of ESN + SPI (+ means concatenation).

The 32bit MSB of the ESN are stored in the IPsec Offload context. These bits are not updated by the device, therefore updates by software are required. They can be performed using `MODIFY_GENERAL_OBJ` command. To coordinate the SW updates with the device, the context also holds the *esn_overlap* field, that indicates how to adjust the ESN when a packet arrives. The fol-

Following table describes how the device uses the ESN MSBs (*esn_msb* field) according to the *esn_overlap* and the packets ESP.sequence number.

Table 806 - IPsec ESN MSB usage

packet ESP.sequence number	<i>esn_overlap</i> = OLD	<i>esn_overlap</i> = NEW
sequence number $< 2^{30}$	<i>esn_msb</i> + 1	<i>esn_msb</i>
$2^{30} \leq$ sequence number $< 3 \times 2^{30}$	<i>esn_msb</i>	<i>esn_msb</i>
$3 \times 2^{30} \leq$ sequence number	<i>esn_msb</i>	<i>esn_msb</i> - 1

When the bottom of the replay protection window (lowest sequence number inside the replay protection window) crosses 2^{31} (half sequence number wraparound) the SW should update the *esn_overlap* to OLD.

When the bottom of the replay protection window crosses 2^{32} (full sequence number wrap-around), the SW should update the *esn_overlap* to NEW, and increment the *esn_msb*.

21.7.8.1 ESN update events

For the Full offload mode, the device can report to the offloaded SW when it is required to update the ESN state (*esn_msb*, *esn_overlap*) for an IPsec Offload context. The event is reported by the Object Change Event. To validate that the event was triggered for ESN update, SW should query the IPsec Offload ASO through QUERY_GENERAL_OBJ, or through Access ASO WQE. The *esn_event_arm* will be 0x0 ESN update is required.

Note: Object Change Event for IPsec Offload context may be triggered for multiple reasons. All reasons should be checked when handling the event.

To arm the IPsec Offload context to trigger an event for the next ESN update, Access ASO WQE should be used to set *esn_event_arm*.

[Internal] TBD: exact fields for this type of update

21.7.9 IPsec Lifetime

IPsec may pose a limitation on the number of packets encrypted by an IPSEC context and key before keys are renegotiated. IPsec offload uses two lifetime thresholds per offload object:

- IPsec hard lifetime - defines the maximum number of packets that can be sent with a given IPsec context (and key). Once exceeded, all subsequent IPSEC operations using this object will fail.
- IPsec soft lifetime - defines the threshold that triggers IPsec rekeying. This threshold is expected to be lower than the hard lifetime.

Both thresholds exceeding can be configured to trigger an OBJECT_CHANGE_EVENT event, to trigger a corrective flow.

The lifetime of an IPSEC offload object is managed by a single counter: `remove_flow_packet_count`. This counter should be decremented per packet that has passed the IPsec action (unless replay-protection has failed). This counter is initialized by object creation or by `ACCESS_ASO` WQE to the required value of the hard lifetime. When the counter reaches the soft lifetime `remove_flow_soft_lifetime`, an event is triggered (soft). When the counter reaches zero, an event is triggered as well (hard). Once the counter is zero, a corresponding syndrome will be generated for all packets going through the IPsec action. Once an event is triggered (or at initialization), it is required to arm the event(s) to receive a new event when a threshold is exceeded by setting `soft/hard_lifetime_arm` in the IPsec offload object. To verify which threshold has exceeded, a query of the object may be done using `ACCESS_ASO` WQE.

21.7.9.1 Lifetime update Example

The following is an example of `ASO_ACCESS` WQE for updating `remove_flow_soft_lifetime`.

Ctrl Segment:

- `OPCODE` - `ACCESS_ASO`
- `OP_MOD` - `IPSEC`
- `ctrl_general_id` - IPSEC Offload object handle
- WQE size - 0x8 (Ctrl seg. and ASO CTL seg.)

ASO CTL segment:

- `data_mask_mode` - `BITWISE_64BIT`
- `Condition_0_enum` - `ALWAYS_TRUE`
- `condition_0_offset` --> 3 (`remove_flow_soft_limit` location)
- Bitwise data - {reserved 32 bit, new `remove_flow_soft_limit`}
- Bitwise mask - 0x0000 ffff.

21.7.9.2 [Internal] Supporting Lifetime values Larger than 32 bit

It is possible to configure a value larger than 2^{32} in the IPsec context lifetime by the following routine:

Pseudo code:

Input:

- `hard_lifetime` – 64bits
- `soft_lifetime` – 64bits

Note: Verify that `soft_lifetime < hard_lifetime`.

Initialization Function:

- Save `hard_lifetime` and `soft_lifetime`:
 - `real_hard_lifetime=hard_lifetime;`

- `real_soft_lifetime=soft_lifetime;`
- If `hard_lifetime >= 2^32` and `soft_lifetime >= 2^32`:
 - Set the counter X to $(2^{31} + \text{hard\_lifetime} \% 2^{31})$, and the comparator C1 to 2^{31} .
 - `last_x=(2^31 + hard_lifetime % 2^31);`
- If `hard_lifetime > 2^32` and `soft_lifetime < 2^32`:
 - Set the counter X to $(2^{31} + \text{hard\_lifetime} \% 2^{31})$, and the comparator C1 to `soft_lifetime`.
 - `last_x=(2^31 + hard_lifetime % 2^31);`
- If `hard_lifetime < 2^32` and `soft_lifetime < 2^32`:
 - Set the counter X to `hard_lifetime`, and the comparator C1 to `soft_lifetime`.
 - `last_x=hard_lifetime;`

Update Function:

After the soft lifetime event is triggered, act according to the value of `real_soft_lifetime` (soft_lifecycle expired or 2^{31} comparator notification)

- If `real_soft_lifetime > 2^31`:
 - Atomically read-modify-write to the counter X (`read_x = atomic_read(X);`)
 - `real_soft_lifetime -= (last_x - read_x);`
 - `real_hard_lifetime -= (last_x - read_x);`
 - `Atomic_set_bit(X, 2^31);`
- If `real_soft_lifetime <= 2^31` and `real_hard_lifetime > 2^31`:
 - Atomically read-modify-write to the counter X (`read_x = atomic_read(X);`)
 - `Atomic_set_bit(X, 2^31);`
 - `Atomic_set(C, real_soft_lifetime)`
- If `real_soft_lifetime <= 2^31` and `real_hard_lifetime <= 2^31`:

21.7.10 `Atomic_set(C, real_soft_lifetime)` IPsec Syndromes

After the IPsec Decryption action a syndrome is returned in a the “IPsec syndrome”. After any Full offload operation, a metadata register will hold an additional syndrome “ASO return value”.

Packets subject to IPsec offload in can take different actions according to the those syndromes (For Crypto offload Encryption there is no applicable syndrome). By steering according to the syndrome the operation success can be validated, failures can be monitored and exception packets can be passed to SW for logging and auditing.

Note: `ipsec_syndrome` can only be used for matching after a Decryption operation. In addition it cannot be modified by Header Modify operation. It may be copied into a different field/header.

The following table describes the syndrome meaning:

Table 807 - IPSEC Syndromes Description

Syndrome	location	Description
IPsec	ipsec_syndrome	Decryption operation status: 0x0: IPSEC_OK - packet decrypted and authenticated successfully 0x1: ICV_FAIL - Authentication failed 0x2: BAD_TRAILER - Trailer length exceeded ESP payload All other values indicate undefined behavior, assume that the decryption and/or authentication have failed, and that the packet is corrupted. other: unknown
IPsec ASO	ASO return register0 ^a [7:0]	Full-offload validation status: 0x0: ASO_OK - packet authenticated successfully 0x1: BAD_REPLAY - Replay protection failed (Decryption flow) 0x2: HARD_LIFETIME - IPsec offload context reached its hard lifetime threshold. All other values indicate undefined behavior, assume that the replay protection has failed.

- a. The ASO return register defines 2 metadata steering registers of type C. The exact register IDs are selected by the IPsec Offload context. The even register will get the syndrome while the odd register will get the ESP.sequence number

21.7.11 Limitations

1. IPSEC offload is not supported for IPoIB
2. IPsec offload is not supported for fragmented IP packets

Note: According to the RFC4301 IPsec transport mode must never be used when plaintext IP fragments can be sent/received.

21.7.12 IPsec Decryption for unsupported packets

For cases where a packet cannot perform the IPsec Decryption offload, SW may continue using some HW offload, while handling the parts of the flow in SW. SW must decrypt and authenticate the packet (using its own copy of ESN) and then perform replay protection (and update hard and soft limit counter) through the device.

1. Check if packet is new enough to move window forward:
 - a. Condition: packet.sequence_number > replay protection window high bound
 - b. Action1: set replay protection window high bound with packet.sequence_num
 - c. Action2: set replay protection window bit of the packet.sequence_number replay bit to 1. Clear replay protection window bits between the old and new replay protection window high bound.

2. If the packet is new enough, then we are done. Otherwise, check whether the packet is older than the window, or inside window and has not been accepted yet:
 - a. Condition1: $\text{packet.sequence\_number} + \text{Window size} \geq \text{replay protection window high bound}$
 - b. Condition2: $\text{replay protection window}[\text{packet.sequence\_number}] = 0$
 - c. Action: set replay bit

The IPsec ASO holds the replay protection window high bound in the field ***mode_parameter***. This field represents the highest sequence number in the replay protection window. The ***replay_protection_window*** is a cyclic presentation of the window, the MSB of the window is selected according to the LSBs of the replay protection window high bound (***mode_parameter***).

TBD: need an example

The replay protection flow described above is done using the ASO Access WQE, which provides an atomic access to the IPsec ASO context, a conditional update (using up to 2 conditions) and optionally scatters the context to the memory with values prior to the update.

The first WQE checks if the packet is actually newer than window and must advance it. In this case the field replay protection window high bound must be advanced. In addition, the replay protection window must be zeroed for all sequence numbers between the old and new replay protection window high bound. The second WQE is required only if the first WQE conditions have failed. In this case either the packet is within the window or too old. Two conditions verify these, and the relevant bit is set if the packet is indeed within window. If the second WQE conditions failed, the packet is either already accepted or outside the window and therefore should be dropped.

To verify the success or failure of the WQE conditions, the SW should use the IPsec ASO context scattered to memory as a result of the WQE, and check the same conditions as the WQE has done.

TBD: part of lifetime feature

[Internal] In addition, the soft and hard lifetime counters must be decremented. For handling the soft/hard lifetime see Soft/Hard Lifetime Events.

21.7.12.1Replay Protection Window High Bound Conditional Update

Following is an example of an ASO Access WQE performing the first check for replay protection as previously described.

Table 808 - Conditional Update Window High Bound WQE example

Segment	Field	Value	Details
Ctrl	OPCODE	ACCESS_ASO	generic for all IPsec ASO access WQEs
	OPC_MOD	IPSEC	
	ctrl_general_id	Ipsec Offload object ID	
	Other fields in this segment should be filled according to normal guidlines of WQE construction		

Table 808 - Conditional Update Window High Bound WQE example

Segment	Field	Value	Details
ASO Ctrl	va_h	Memory location to which the current value of the context will be scattered to	Indicate device to scatter the IPsec ASO to memory, so the result of the condition could be calculated by SW
	va_l		
	l_key		
	read_enable	0x1	
	data_mask_mode	CALCULATED_64BYTE	Mask should be calculated by the device since a single bit needs to be updated in the <i>replay_protection_window</i> in addition to the replay protection window high bound (<i>mode_parameter</i>), which cannot be both covered by a single WQE otherwise.
	condition_0_operand	CYCLIC_GREATER	
	condition_0_offset	0x8	the offset of the replay protection window high bound (<i>mode_parameter</i>) in the IPsec ASO
	condition_0_data	Packet.ESP.sequence_number	
	condition_0_mask	0xFFFF_FFFF	Indicate the entire field should be included
	condition_1_operand	ALWYAS_TRUE	Disable condition 1 as it is not required
	condition_operand	AND	
	data_mask	{28'd0, 4'b1111, 32'd0}	mark only the replay protection window high bound (<i>mode_parameter</i>). The calculated mask will add the specific bit in replay protection window to the mask
Other fields are reserved			
ASO Data	bytewise_data	{228'd0, ESP sequence number[31:0], 256'd1 << Packet.ESP.sequence_number}	In case of condition success, the new replay protection window high bound (<i>mode_parameter</i>) is the current packet sequence number, and the appropriate bit in replay protection window should be 1.

21.7.12.2Replay Protection Window conditional update

Following is an example of an ASO Access WQE performing the second check for replay protection as previously described. SW should wait for the first WQE to complete, check if the condition has succeeded, and only if not post the second WQE.

Table 809 - Conditional update Replay Protection Window WQE example

Segment	Field	Value	Details
Ctrl	OPCODE	ACCESS_ASO	generic for all IPsec ASO access WQEs
	OPC_MOD	IPSEC	
	ctrl_general_id	Ipsec Offload object ID	
	Other fields in this segment should be filled according to normal guidlines of WQE construction		

Table 809 - Conditional update Replay Protection Window WQE example

Segment	Field	Value	Details
ASO Ctrl	va_h	Memory location to which the current value of the context will be scattered to	Indicate device to scatter the IPsec ASO to memory, so the result of the condition could be calculated by SW
	va_l		
	l_key		
	read_enable	0x1	
	data_mask_mode	BITWISE_64BIT	
	condition_0_operand	CYCLIC_GREATER	
	condition_0_offset	0x8	the offset of the replay protection window high bound (<i>mode_parameter</i>) in the IPsec ASO
	condition_0_data	Packet.ESP.sequence_number + replay protection window size	
	condition_0_mask	0xFFFF_FFFF	Indicate the entire field should be included
	condition_1_operand	EQUAL	
	condition_1_offset	(packet.ESP.sequence_number % window size)/32	The relevant DW offset of the current sequence number inside the replay protection window
	condition_1_data	32'd0	
	condition_1_mask	32'd1 << packet.ESP.sequence_number	only the relevant bit in the replay protection window should be compared
	condition_operand	AND	
	data_offset	(packet.ESP.sequence_number % window size)/64	The relevant double-DW offset of the current sequence number inside the replay protection window
	data_mask	32'd1 << packet.ESP.sequence_number[5:0]	Set the relevant bit in replay protection window to 1
	bitwise_data	32'd1 << packet.ESP.sequence_number[5:0]	

22 CAPI- Coherent Accelerator Programming Interface

22.1 Overview

The HCA can leverage the IBM CAPI technology to implement address translation similar to the PCI-SIG PASID extension. Using CAPI, the HCA can use the host page tables rather than the Mellanox specific address translation mechanisms. The feature is similar in nature and user facing value to the on-demand paging feature, which is host independent. However, CAPI can be more efficient in some scenarios due to page tables sharing. The sharing removes the need to maintain NIC specific translation tables and allows the HCA to access pages that the software touched on the host CPU without triggering another page fault upon HCA access.

The HCA implements CAPI as a second level of memory mapping, similar to the way the HCA interacts with an IOMMU. When CAPI is enabled, The HCA can Enable CAPI translation for certain MKeys. When such an MKEY is used, the regular bounds checking and translation are followed by a CAPI translation. For example, an RDMA transaction will provide an RKey and an RDMA-VA. The HCA will lookup an MKey according to the RKey from the wire. The HCA will then perform a regular MKEY translation according to the MKEY type. Assuming CAPI is enabled on this MKEY, the HCA will then use the CAPI mechanism to translate the address it got in the previous stage to obtain to final address to use.

22.2 CAPI Address Translation

CAPI uses a process element Identifier PE-ID to identify an address space. The HCA follows the PCI-SIG definition of a process address space identifier (PASID) to identify an address space. A notable difference between ATS and CAPI is that ATS requires only the address and the PASID to obtain a translation, while CAPI needs additional information. Consequently, in CAPI mode, the HCA does another lookup based on the PASID value to obtain a CAPI process element context (PEC) where the relevant information is stored. CAPI PECs are created using the CREATE_PEC command.

22.3 CAPI Owner

Due to CAPI limitations, all the CAPI invalidation must be serviced by a single function. The HCA designates the first function to enable CAPI as the CAPI owner and services all invalidations through this function. All the other functions depend on the CAPI owner and as a result performing DISABLE_HCA on the CAPI owner triggers an FLR on all the active CAPI functions.

22.4 CAPI Initialization Flow

CAPI should be enabled after validation of the FW version and before writing the physical location of the command queue address to the initialization segment.

The CAPI initialization flow on the HCA is as follows:

- Check that the HCA is CAPI capable by running ICMD_QUERY_CAPABILITY.capi==1. [See TBD](#)
- Set owner field in CAPI_CTRL using ICMD_ACCESS_REGISTER command.

- Query CAPI_CTRL register using ICMD_ACCESS_REGISTER to get owner field.

If the owner field is set:

- Obtain the CAPI host parameters from the host OS.
- Use ICMD_ACCESS_REGISTER command to write the CAPI host parameters to the HCA while setting the *enable_invalidations* field.
- Poll the *enable_invalidations* field using ICMD_ACCESS_REGISTER until it is set
- Enable CAPI on the host.
- Set *enable_invalidations* and *enable_translations* in CAPI_CTRL using ICMD_ACCESS_REGISTER command.
- Poll the *enable_invalidations* and *enable_translations* fields using ICMD_ACCESS_REGISTER until they are both set.

22.5 CAPI Teardown Flow

If the owner field is set:

- Clear *enable_translations* in CAPI_CTRL using ICMD_ACCESS_REGISTER command to write the
- Poll the *enable_invalidations* field using ICMD_ACCESS_REGISTER until it is set.

Clear *enable_invalidations* and *enable_translations* in CAPI_CTRL using ICMD_ACCESS_REGISTER command.

23 Tunneled Operations

This sections is only applicable to Power 9 systems.

Power 9 supports tunneling of Powerbus commands over PCIe. Tunneled operation are sent as regular PCIe writes from the HCA toward the host. The address and payload of the PCIe write are defined by IBM. The host detects those operation based on the MSB bits of the address

According to `asn_match_value` and `asn_match_mask`. The host responds to a tunneled operation with a regular PCIe writes toward a special response bar on the HCA.

To enable the of tunneled operation, SW needs to perform `SET_TUNNELED_OPERATION_PARAMS` in the initialization flow (See Section 7.2, “HCA Driver Start-up,” on page 244)and provide the tunneled operations parameters (MSB `asn_match`, response address....).

The HCA supports two types of tunneled operation:

- AS Notify
- Tunneled atomic

23.1 AS_Notify

This feature is supported only when `.as_notify==1`.

AS_Notify is low-latency HW based thread wakeup mechanism.

Instead of actively polling a CQ, the processor can arm the CQ and issue a “wait-a-sec” instruction to put the HW thread to sleep. Putting an HW thread to sleep, prevents the HW thread from consuming resources, improves the performance of other HW threads that share the core and saves power.

These same benefits can be accomplished using “sleep” and interrupts. However, interrupt based mechanisms result in an expensive and high latency context switch which is undesirable.

The Power 9 support a tunneled operation called `as_notify` that can be used to wake up a specific software thread identified by a Process Element Entry (LPID:PID:TID).

The response to an `as_notify` operation is success if the relevant software thread is currently executing on some HW thread and failure otherwise.

The feature is enabled per CQ when it is created by setting `as_notify` and process parameters(LPID:PID:TID). If the CQ is armed the HCA will try to issue an `as_notify` to the relevant thread instead of generating an EQE. If the `as_notify` fails due to lack of resources on the HCA or if the `as_notify` response indicates that the thread is not running, the HCA will fall back to the regular flow of posting an EQE to the EQ associated with the CQ.

Note that the HCA does the `as_notify` in a best effort manner, so an EQE might be received even if the relevant thread is actually waiting on a “wait” instruction. Therefore upon reception of such

an EQE the Software should take measures to wake up the relevant thread. Otherwise that thread might remain in the “wait” condition for a nondeterministic amount of time.

23.2 Tunneled atomic:

This feature is supported only when `.tunneled_atomic==1`.

TBD - add description

24 Extended Atomic Operations

Masked semaphore operations introduced in ConnectX® are extended to device supporting arguments within the range of 1 to 256 bytes. The device will execute atomic operations on an entire argument. Arguments must be naturally-aligned in host memory on the destination (semaphore) side.

Atomic operations supported are Compare&Swap and Fetch&Add. *MASK* argument used to mask semaphores in Compare&Swap operation and to identify semaphores' boundary in Fetch&Add operation.

The device uses same wire protocol (opcode) as ConnectX®-3 for masked atomic operation. The size of the operand is implied by the size of packet.

The device will provide atomicity between any atomic operations who have some addresses overlapping.

Not-masked atomic operations are only 8 byte atomics. All extended atomics operations include 'mask'.

24.1 Masked Compare and Swap (MskCmpSwap)

The MskCmpSwap atomic operation is an extension to the CmpSwap operation defined in the IB spec. MskCmpSwap allows the user to select a portion of the target data for the "compare" check as well as to restrict the swap to a (possibly different) portion. The pseudocode in [Figure 107](#) describes the operation.

Figure 107: MskCmpSwap Pseudocode

```

atomic_response = *va
if (((cmp XOR *va) AND cmp_mask) is ZERO) then
    *va = (*va AND NOT(swap_mask)) OR (swap AND
swap_mask)
return atomic_response

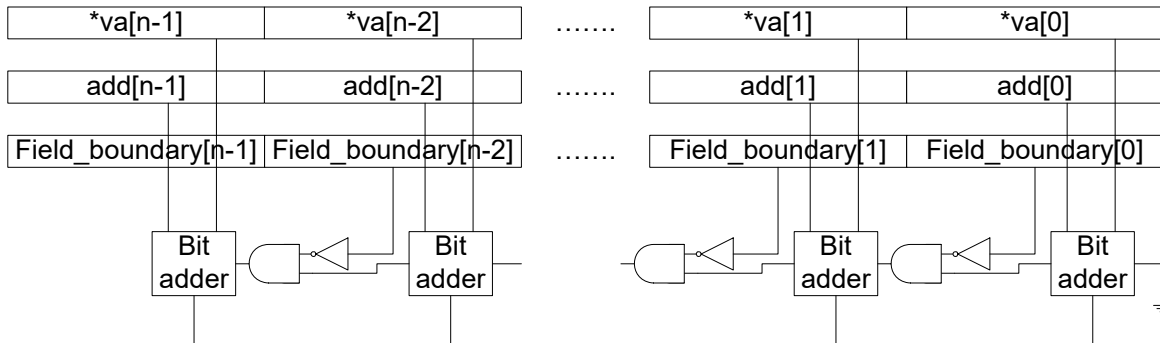
```

The additional operands are carried in the Extended Transport Header. Atomic response generation and packet format for MskCmpSwap is as for standard IB Atomic operations.

24.2 Multifield Fetch and Add (MFetchAdd)

The MFetchAdd Atomic operation extends the functionality of the standard IB Fetch&Add by allowing the user to split the target into multiple fields of selectable length. The atomic add is done independently on each one of these fields. A bit set in the field_boundary parameter specifies the field boundaries. A 64-bit boundary is implicit (carry cannot cross 64-bit boundary of the operand). [Figure 108](#) describes the operation.

Figure 108: MFetchAdd

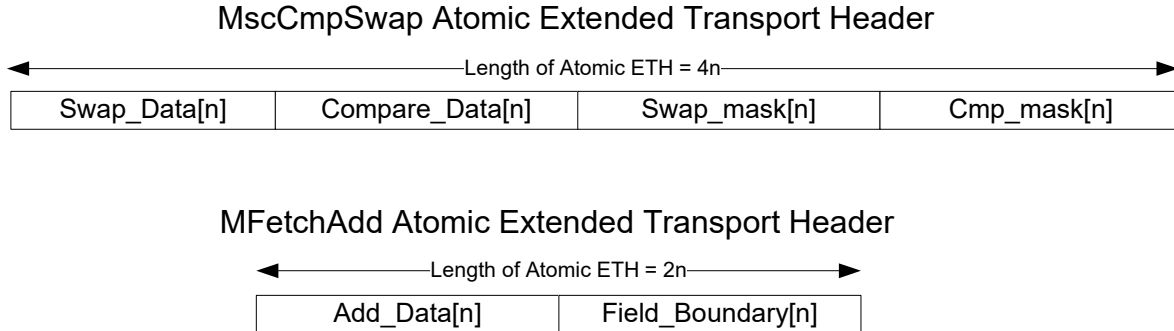


The additional operands are carried in the Extended Transport Header. Atomic response generation and packet format for MFetchAdd is the same as for standard IB Atomic operations.

24.3 Packet Format

All operands of the extended atomic operation are carried in the Atomic Extended Transport Header. The size of this header determines the size of the operand. Layout of Atomic Extended Transport Header is shown in [Figure 109](#).

Figure 109: Extended Atomics operation - Atomic Extended Transport Header



Atomic response is delivered in the payload of the Atomic Acknowledge packet, carrying return argument in the packet payload.

Extended atomic opcodes are:

- Masked Compare and Swap - 0x18
- MFetchAdd - 0x19

24.3.1 Short Atomics (1,2 or 4 bytes)

A 4-byte extended atomic packet ETH is 32 bits, as shown in [Figure 109](#), “Extended Atomics operation - Atomic Extended Transport Header”. Opcodes are the same as for the larger extended atomic operation.

Atomic arguments of less than 4 bytes are supported by using 4-byte arguments with appropriate masks. The semaphore length is interpreted by mask value - non-masked part of the 32-bit argument forms a single naturally-aligned semaphore of 1, 2 or 4 bytes. Responder will always read 4 bytes, execute atomic operation and write back unmasked part of the argument. Response message will carry 32-bit value with semaphore value returned in the unmasked part of this argument.

[Table 810, “Masks for 1B and 2B atomic operation,” on page 1041](#) shows the masks for 1B and 2B atomic operations. Let A be a 4-byte aligned address. It is the remote address in the atomic packet. Compare data, swap data and data location is aligned to the location of the FF’s in the masks.

Table 810 - Masks for 1B and 2B atomic operation

	A	A+1	A+2	A+3
C&S 1B	C-mask = 0xFF000000 S-mask = 0xFF000000	C-mask = 0x00FF0000 S-mask = 0x00FF0000	C-mask = 0x0000FF00 S-mask = 0x0000FF00	C-mask = 0x000000FF S-mask = 0x000000FF
C&S 2B	C-mask = 0xFFFF0000 S-mask = 0xFFFF0000	NA	C-mask = 0x0000FFFF S-mask = 0x0000FFFF	NA
SWAP 1B	C-mask = 0x00000000 S-mask = 0xFF000000	C-mask = 0x00000000 S-mask = 0x00FF0000	C-mask = 0x00000000 S-mask = 0x0000FF00	C-mask = 0x00000000 S-mask = 0x000000FF
SWAP 2B	C-mask = 0x00000000 S-mask = 0xFFFF0000	NA	C-mask = 0x00000000 S-mask = 0x0000FFFF	NA
F&A 1B	Data = 0XX000000 mask = 0x80000000	Data = 0X0XX0000 mask = 0x00800000	Data = 0x0000XX00 mask = 0x00008000	Data = 0x000000XX mask = 0x00000080
F&A 2B	Data = 0XXXXX0000 mask = 0x80000000	NA	Data = 0x0000XXXX mask = 0x00008000	NA

24.4 WQE Format

WQE format of extended atomic operation is shown in [“Send WQE Format” on page 310](#).

24.5 RDMA Write Atomicity

The device supports atomicity between RDMA WRITE operations as well as atomicity between them and atomic operations. This is to allow semaphore releasing by write operation.

For WRITE operation to be handled in atomic manner (with respect to atomic operations and other “atomic writes”), the following should hold:

- The receive QP on the write target should be enabled for atomic-writes
- The size of the write operation should not exceed the “max_atomic_size” configuration of the HCA

The write operation should not cross the “max_atomic_size” natural alignment boundaries.

25 Remote Micro-Applications

The device introduces supports Remote Micro-Applications, which enables the issuing of various micro-applications over the network, to be performed on a remote HCA. The support of Remote Micro-Applications is reported by HCA_CAP.atomic.atomic_operations (See Table 1577, “Atomic Capabilities Layout,” on page 1728).

Remote Micro-Applications are meant to execute operations in the RDMA responder.

The Remote Micro-Applications atomics are defined to perform the arithmetic operation in the responder according to the remote host Endianness.

Two Remote Micro-Applications flavors are defined:

- **LOAD Remote Micro-Applications** – returning response to the requester, where the response is a Read response.
- **STORE Remote Micro-Applications** – terminating in the responder side, where the response is an ACK.

The return value for the LOAD Remote Micro-Applications atomics is the original value (and boundary/id when relevant). See Table 811, “Remote Micro Application Codes Summary,” on page 1042.

The supported Operand sizes are of 4B and 8B, which is reported per *operation_code* in HCA_CAP.Atomic.load_remote_micro_app_4byte, load_remote_micro_app_8byte, store_remote_micro_app_4byte and store_remote_micro_app_8byte. See Table 1577, “Atomic Capabilities Layout,” on page 1728.

Table 811 - Remote Micro Application Codes Summary

Operationcode name	operation_code	Number of operands	number of masks	response size	rrDescription
swap_masked	0x1	1	1	4B,8B	Masked swap of memory content with the operand
cas_e_masked	0x2	2	2	4B,8B	Masked compare and Swap of memory content if memory content is equal to the first operand, by the second operand
cas_ne_masked	0x3	3	2	4B,8B	Masked compare and Swap of memory content if memory content is not equal to the first operand, by the second operand
add	0x4	1	0	4B,8B	Atomic sum of memory content with operand
AND_masked	0x5	1	1	4B,8B	Bitwise AND Masked of operand with memory content
XOR_masked	0x6	1	1	4B,8B	Bitwise XOR Masked of operand with memory content

Table 811 - Remote Micro Application Codes Summary

Operationcode name	operation_code	Number of operands	number of masks	response size	rrDescription
OR_masked	0x7	1	1	4B,8B	Bitwise OR Masked of operand with memory content
unsigned max	0x8	1	0	4B,8B	Replace memory content with the unsigned max of memory content and operand
signed max	0x9	1	0	4B,8B	Replace memory content with the signed max of memory content and operand
unsigned min	0xA	1	0	4B,8B	Replace memory content with the unsigned min of memory content and operand
signed min	0xB	1	0	4B,8B	Replace memory content with the signed min of memory content and operand
unsigned max-id_masked	0xC	2	2	8B,16B	Perform an unsigned masked max on memory content and the first operand. if the masked operand is larger, replace memory content with the first operand and the masked id (second operand)
signed max-id_masked	0xD	2	2	8B,16B	Perform a signed masked max on memory content and the first operand. if the masked operand is larger, replace memory content with the first operand and the masked id (second operand)
unsigned min-id_masked	0xE	2	2	8B,16B	Perform an unsigned masked min on memory content and the first operand. if the masked operand is smaller, replace memory content with the first operand and the masked id (second operand)
signed min-id_masked	0xF	2	2	8B,16B	Perform a signed masked min on memory content and the first operand. if the masked operand is smaller, replace memory content with the first operand and the masked id (second operand)
add bounded (<=), boundary in addr+s	0x10	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA]+Operand1) <=Memory[VA+S] { Memory[VA] = Memory[VA] +Operand1 }

Table 811 - Remote Micro Application Codes Summary

Operationcode name	operation_code	Number of operands	number of masks	response size	rrDescription
add bounded (>=), boundary in addr+s	0x11	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA]+Operand1) >=Memory[VA+S] { Memory[VA] = Memory[VA] +Operand1 }
add equal (=), boundary in addr+s	0x12	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA]+Operand1) ==Memory[VA+S] { Memory[VA] = Memory[VA] +Operand1 }
add bounded (<=), boundary in addr,	0x13	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA+S]+Operand1) <=Memory[VA] { Memory[VA+S] = Memory[VA+S] +Operand1 }
add bounded (>=), boundary in addr,	0x14	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA+S]+Operand1) >=Memory[VA] { Memory[VA+S] = Memory[VA+S] +Operand1 }
add equal (=), boundary in addr,	0x15	2 (second is reserved)	0	8B,16B	S= operand size (4 or 8 Byte) If(Memory[VA+S]+Operand1) ==Memory[VA] { Memory[VA+S] = Memory[VA+S] +Operand1 }

26 Global Atomicity Guaranteeing (GAG)

Network atomics are atomics operations guaranteeing atomicity over the network. Such atomics will not prevent the local host from accessing the related memory addresses and break atomicity.

GAG atomics are sent as an atomic operations over PCIe and preserve atomicity with respect to accesses made by the local host as well.

PCIe atomics enablement process is according to PCIe spec and before the driver initialization.: If the atomic enablement is changed after the driver is loaded, the device initiates a FLR. Some of the atomics defined as IB atomics, extended IB atomics or remote micro-applications atomics can be executed as GAG atomics. Each atomic type listed in Table 812 can be executed as GAG atomics only if it's opcode and size is supported as reported in Atomic Capabilities Table, beside the appropriate pci atomic operation and size capabilities (pci_atomic_operation and pci_atomic_size) in Atomic Capabilities Table, See Table 1577, "Atomic Capabilities Layout," on page 1728.

Table 812 - Global Atomicity Guaranteeing (GAG) Summary

PCIe Atomic	Group	Note
FetchAdd 4 Byte	Extended atomic	Masked fetch & add without an effective masking
	Remote micro app	Add
FetchAdd 8 Byte	Extended atomic	Masked fetch & add without an effective masking
	Remote micro app	Add
Swap 4Byte	Extended atomic	Masked CAS with full comparing mask and without an effective masking
	Remote micro app	swap_masked without an effective masking
	Remote micro app	case_e_masked with full comparing mask and without an effective masking
Swap 8Byte	Extended atomic	Masked CAS with full comparing mask and without an effective masking
	Remote micro app	swap_masked without an effective masking
	Remote micro app	case_e_masked with full comparing mask and without an effective masking
compare and swap 4Byte	Extended atomic	Masked CAS without an effective masking
	Remote micro app	case_e_masked without an effective masking

Table 812 - Global Atomicity Guaranteeing (GAG) Summary

PCIe Atomic	Group	Note
compare and swap 8Byte	Extended atomic	Masked CAS without an effective masking
	Remote micro app	case_e_masked without an effective masking
compare and swap 16Byte	Extended atomic	Masked CAS without an effective masking

Part 3: Command Reference and Registers

27 Debug Enhancements

27.1 Dropped Packet Logging

The device can log part of dropped packet. Once a received packet is dropped, the first 128B of it are logged by the device and an event is sent to SW. See “DROPPED_PACKET_LOGGED Event” on page 487 for details. SW can retrieve the logged part of the packet using the GET_DROPPED_PACKET_LOG command. See Section 29.39, “GET_DROPPED_PACKET_LOG,” on page 2220 for details.

The device can log a single dropped packet at a time. Once the packet log is read by the GET_DROPPED_PACKET_LOG command, the device can log another dropped packet.

This mechanism is enabled when the `DROPPED_PACKET_LOGGED` is mapped to an EQ.

27.2 Health Buffer

Table 813 - Health Buffer Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
assert_var0																																00h
assert_var1																																04h
assert_var2																																08h
assert_var3																																0Ch
assert_var4																																10h
																																14h-1Ch
assert_existptr																																20h
assert_callra																																24h
																																28h-2Ch
fw_version																																30h
hw_id																																34h
rtt																																38h
irisc_index								synd								ext_sync																3Ch

Table 814 - Health Buffer Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	assert_var0	When synd==FW_INTERNAL_ERR This is the 1st variable passed to assert	RO
04h	31:0	assert_var1	When synd==FW_INTERNAL_ERR This is the 2nd variable passed to assert	RO
08h	31:0	assert_var2	When synd==FW_INTERNAL_ERR This is the 3rd variable passed to assert	RO
0Ch	31:0	assert_var3	When synd==FW_INTERNAL_ERR This is the 4th variable passed to assert	RO
10h	31:0	assert_var4	When synd==FW_INTERNAL_ERR This is the 5th variable passed to assert	RO
14h-1Ch	96	reserved		
20h	31:0	assert_existptr	When synd==FW_INTERNAL_ERR this is instruction pointer when assert was triggered	RO
24h	31:0	assert_callra	When synd==FW_INTERNAL_ERR This is the next return address after triggering assert	RO
30h	31:0	fw_version	Current FW version	RO
34h	31:0	hw_id	Current device ID	RO
38h	31	rfr	Recovery Flow Required. When set, the error indicated in the health buffer cannot be recovered by the device without a flow involving a reset.	RO
3Ch	31:24	irisc_index	Index of IRISC which caused fatal error	
	23:16	synd	See health_syndrome in Table 15, “Initialization Segment Field Descriptions,” on page 235	
	15:0	ext_synd	Extended syndrome: meaning differs per syndrome. For 0xB - HW_FATAL 0x1 - DPS_BUFFER_OVERRUN 0x2 - ICM fetch page not present error 0x3 - EQ invalid for others, reserved	

In order for SW to confirm FW health, the initialization segment exposes a health counter, syndrome and a related buffer containing advanced debug information.

The health counter is incremented by FW every 10msec and monitored by SW. It should be monitored by an independent thread that is not blocked by waiting for command completion.

If the counter fails to increment it indicates a bad system health state in which case:

- if `health_syndrome` field is 0x0 - this indicates that FW is unable to respond to initialization segment reads and health buffer should not be read.
- if `health_syndrome` field has a value greater than 0x0 - advanced debug information should be read from health buffer.

Following reset, health buffer can be read to ascertain whether an error was encountered prior to reset (for instance, an FFSE error - see [Table 688, “Data Integrity Detection and Reporting,” on page 877](#)). Health buffer should then be cleared to avoid reading stale data after subsequent resets.

27.3 Tracer

The Device Tracer is a mechanism for the device to log important events during its operation. The log includes information about the entire device behavior, not specific for a port or a function, therefore it is available for privileged entities only.

27.3.1 Tracer Ownership

Multiple device drivers may be privileged to configure and activate the device tracer log. Each such driver should attempt to acquire ownership over the tracer by setting `trace_owner` field in “MTRC_CAP -Tracer Capability Register”. If ownership was granted, the same field will be set in the register. If ownership was not granted, the driver may either:

- Periodically reattempt to acquire ownership.
- Register to “Driver Tracer Event”, to receive notification of tracer ownership changes. The driver should reattempt to acquire ownership whenever the event is triggered.

To release the ownership, `trace_owner` field should be cleared. The tracer owner, should not attempt to reacquire ownership.

Tracer ownership may be unexpectedly lost. In such event, traces will not be logged anymore. The driver may attempt to reacquire ownership.

27.3.2 Tracing to Memory

The device tracer log is written to the host memory, into a cyclic buffer. The buffer should be mapped and registered to the device using the method described in [<TBD cross reference to chapter “Address Translation and Protection”>](#), with the following restrictions:

- The buffer size should not exceed the value indicated by `log_max_trace_buffer_size` field in “MTRC_CAP -Tracer Capability Register”.
- The memory region must not use Indirect MKeys
- The memory region must not have signature operations
- The memory region must not be UMR capable
- The memory region must not be invalidated (local/remote)
- The memory region must have Local Write permissions
- The region start address and length must be aligned to 4KB

- The region start address and length must be aligned to 4KB

The support for this mode of tracing is indicated by *trace_to_memory* field in “[MTRC_CAP - Tracer Capability Register](#)”. The tracing mode, the buffer size and the registered MKey should be configured for in *trace_mode*, *log_trace_buffer_size* and *trace_mkey* fields in “[MTRC_CONF - Tracer Configuration Register](#)” respectively.

The device logs traces in blocks of 256B. The last 8B hold a Timestamp trace event (see “[Timestamp Event Traces](#)” on page 1053). Reading this location in the currently processed block, and comparing it to the timestamp of the previous block, will indicate that the new block was completely written to the buffer.

A valid block should be processed according to “[Traces Chronology](#)” on page 1052 and “[Parsing Traces](#)” on page 1053. The method for preventing processing the block without the risk of the device overwriting the buffer is outside the scope of this document. A possible flow may be to copy the block to a different buffer, and validate that the timestamp at the end of the previous block remained the same since that block was processed. If a block is suspected to be overwritten, the driver may chose to discard the traces within it. In any case, it may move to polling the next block.

The last valid timestamp may be assumed to be a timestamp queried by when starting the trace.

Note: All timestamp comparisons should be done in a cyclic matter

The driver may be informed of new Tracers in the buffer by:

- Periodically polling the last timestamp of the current block, as described above
- Register to “[Driver Tracer Event](#)”, to receive notification that the buffer is not empty.

When using the Device Tracer Event, the driver should arm the event by setting *arm_event* in “[MTRC_CTRL - Tracer Control Register](#)”. A single event will be triggered once the device detects traces were logged since the last Device Tracer Event. Once an event is triggered, an additional event will not be triggered until the event is re-armed. The event may be armed regardless of tracer current active status. An armed event is not unarmed until the event is fired, or the tracer ownership is released.

Note: When mixing polling and event driven processing it may happen that a Device Tracer Event will be received when all traces in the buffer were already processed.

Note: When using event driven processing it may happen that a Device Tracer Event will be received when all traces in the buffer appear to be invalid. In such case, the Event should be rearmed without any processing of events.

27.3.3 Traces Chronology

An important property of a trace, is the time it was logged. The device logs a Timestamp Event Trace, reporting the current device time. The device time is measured in cycles., which can be translated into real time following the same process described in [Section 8.19.12, on page 471](#).

Each other type of trace holds a partial timestamp, representing the time it was logged. This time is given in the same units and allows calculating the exact time the trace was logged, in conjunction with the next Timestamp Event Trace.

```

if (trace.timestamp < next_timestamp_event_trace.timestamp[6:0])
    trace_timestamp = {next_timestamp_event_trace.timestamp[52:7], trace.timestamp}
else
    trace_timestamp = {next_timestamp_event_trace.timestamp[52:7] - 1, trace.timestamp}

```

When parsing event traces, the next Timestamp Event Trace should be searched, to be used for constructing the previous events.

27.3.3.1 Invalid Timestamp Event Trace

When the device is unable to guarantee that the time distance between the Timestamp Event Trace and the previous Event Traces is short enough to construct a viable timestamp, the Timestamp Event Trace will be marked as unreliable. The marking indicates that all previous Event Traces cannot construct a full timestamp. The field **urts_ver** in “[MTRC_CAP -Tracer Capability Register](#)” indicates which field marks the unreliability, **urts_0** or **urts_1**.

27.3.4 Parsing Traces

The logged traces have the following format:

Table 815 - Trace Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lost	timestamp								event_id								event_data[47:32]														00h	
	event_data[31:0]																															04h

Table 816 - Trace Fields

Offset	Bits	Name	Description
00h	31	lost	When set, one or more event traces were dropped and not logged since last logged event trace.
	30:24	timestamp	Elapsed cycles since last Timestamp Event Trace
	23:16	event_id	Identifier of the logged event trace
	15:0	event_data[47:32]	Event-trace-specific data, interpreted according to the event_id
04h	31:0	event_data[31:0]	

Further parsing of the trace is done according to specific type of the trace indicated by **event_id**. The following sections describe the different methods of parsing the trace data.

27.3.4.1 Timestamp Event Traces

This trace is generated at least once every 256B block of traces, to provide a reference to the device time for the previous traces. See “[Traces Chronology](#)”. The Timestamp Event Trace is

identified by *event_id* 0xFF, and has a one of 2 different format, indicated by *trc_ver* in “MTRC_CAP -Tracer Capability Register” :

Table 817 - Timestamp Event Trace version 0 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
timestamp[7:0]								event_id								urts			timestamp[52:40]											00h		
timestamp[39:8]																															04h	

Table 818 - Timestamp Event Trace version 0 Fields

Offset	Bits	Name	Description
00h	31:24	timestamp[7:0]	8 LSB of the elapsed time until the trace was logged, given in device cycles
	23:16	event_id	Timestamp Event Identifier 0xFF
	15	urts	When set, the timestamp of previous event traces cannot be reconstructed using this Timestamp Event Trace, up to the previous Timestamp Event Trace.
	12:0	timestamp[52:40]	45 MSB of the elapsed time until this trace was logged, given in device cycles
04h	31:0	timestamp[39:8]	

Table 819 - Timestamp Event Trace version 1 Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
timestamp[7:0]								event_id										urts	timestamp[52:40]											00h		
timestamp[39:8]																															04h	

Table 820 - Timestamp Event Trace version 1 Fields

Offset	Bits	Name	Description
00h	31:24	timestamp[7:0]	8 LSB of the elapsed time until the trace was logged, given in device cycles
	23:16	event_id	Timestamp Event Identifier 0xFF
	13	urts	When set, the timestamp of previous event traces cannot be reconstructed using this Timestamp Event Trace, up to the previous Timestamp Event Trace.
	12:0	timestamp[52:40]	45 MSB of the elapsed time until this trace was logged, given in device cycles
04h	31:0	timestamp[39:8]	

27.3.4.2 String Traces

These traces are parsed as a group of one or more traces, referred to as trace message. The trace message represents a C-like printf() string and its parameters, for example:

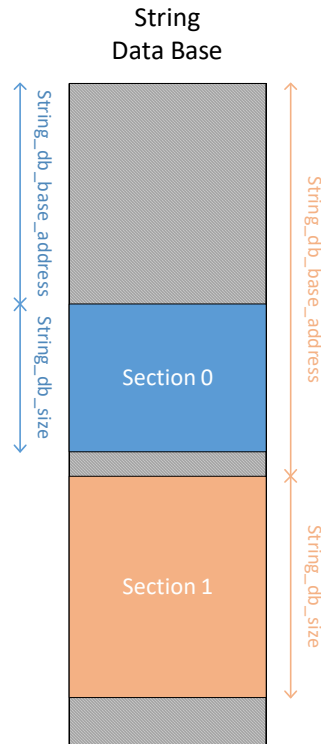
```
printf("Packet received at qp %06x on port %d", qp_num, port)
```

Each trace holds 2 sequence numbers:

- Trace Message Sequence Number (TMSN) - incremented for every message. TMSN counting is independent for different *event_id*.
- Trace DWord Sequence Number (TDSN) - incremented for every trace in the message, starting at 0x0. TDSN counting is independent for different TMSN.

The first trace of a message holds the pointer to the string. The pointer is an offset inside a database holding all the strings produced by this kind of traces. Each string in the database is NULL-terminated. The database may be built by one or more sections. The number of sections is given by *num_string_db* field in “[MTRC_CAP -Tracer Capability Register](#)”. Each section holds a part of the String DB starting at a different offset and holding a different amount of data. The same register holds the start offset and the size of each section in *string_db_param[section].string_db_base_address* and *string_db_param[section].string_db_size* respectively. Each section can be read separately by “[MTRC_STDB - Tracer String DB Register](#)”. Multiple reads of different offsets inside the same section may be required to read the entire section.

Figure 110: String DataBase Layout



Note: The String DB is not necessarily fully utilized. The string sections may not cover the entire address space, and may not be sequential.

The traces of a message, beyond the first one, hold the printf() parameters, in the order of their appearance in the string. All parameters are 32 or 64 bit integers. 64 bit integers are stored in two consecutive traces (sequential TDSNs), the first one (lower TDSN) holding the MSB. The number of parameters in the message can be determined by parsing the string and finding the escape character '%'.
 [Internal] '%%' is not used. 64 bit parameters use only '%llx'

Note: Traces of the same message are not necessarily sequential. Other traces may be logged in between the traces of a message, with a different *event_id*.

The string traces are identified by *event_id* in the range defined by *first_string_trace* and *num_string_trace* fields in “MTRC_CAP-Tracer Capability Register”. Their *event_data* has the following format:

Table 821 - String Trace Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lost	timestamp								event_id								tmsn								tdsn				00h			
string_param																																04h

Table 822 - String Trace Fields

Offset	Bits	Name	Description
00h	31	lost	Some Traces of this type have been triggered after (later) than this one, but could not be logged.
	30:24	timestamp	See Table 816, “Trace Fields,” on page 1053
	23:16	event_id	See Table 816, “Trace Fields,” on page 1053
	15:3	tmsn	Tracer Message Sequence Number (TMSN)
	2:0	tdsn	Tracer DWord Sequence Number (TDSN)
04h	31:0	string_param	for <i>tdsn</i> = 0x0, this field holds the pointer to the string for <i>tdsn</i> > 0x0, this field holds one of the string parameters

27.3.4.2.1 String Severity

The strings provided by the String Traces may indicate their severity by values 0-9, where 0 is the highest severity. The severity is placed in the first character of the string. Strings with severity level 9 should not be printed. Strings with severity level value outside the 0-9 range should be assumed to have the highest severity level.

27.3.4.3 Lost Event Traces

When the device is not able to log a trace, even though the triggering event has happened, the *lost* indication will be set.

27.3.4.4 Unrecognized Traces

For traces that were not successfully parsed, the trace data should be printed unparsed along with the calculated timestamp, lost and *event_id* and provided as raw data for Mellanox technical support, as a 16-digit hexadecimal number, in the following format:

```
<timestamp> <lost> <event_id> 0x<Trace Event>
```

Some reasons for failed parsing may be:

- Unknown *event_id* value
- Missing string in the String DB
- String message terminated without receiving all required parameters

27.3.5 Tracer Initialization

The Device Tracer Initialization should comply with the following flow:

1. Query the device capabilities for supporting Trace Registers [<TBD cross reference to MCAM>](#)
2. Query “MTRC_CAP -Tracer Capability Register” for Tracer capabilities

3. If using Events, map “Driver Tracer Event” to an Event Queue [<TBD cross reference to Event Queue Mapping>](#)
4. Acquire Tracer ownership according to “Tracer Ownership” on page 1051. Once ownership acquired continue to the next step.
5. Query the Strings DB as described in “String Traces” on page 1055
6. Allocate and Register a memory buffer to hold the logged traces,
7. Configure the required parameters to “MTRC_CONF - Tracer Configuration Register”. Tracer parameters can be modified only when the tracer is not active, unless specified otherwise.
8. Start the tracer using “MTRC_CTRL - Tracer Control Register”
9. Arm the Trace Event, If working with in interrupt mode.

Note: If the driver is trying to re-initialize the tracer after unexpected loss of ownership, steps 5 and 6 can be skipped.

27.3.6 Tracer Teardown

When the driver no longer wishes to own the tracer (for example, on driver tear-down), or unexpectedly lost ownership, the driver should comply with the following flow:

1. Release Tracer ownership according to “Tracer Ownership” on page 1051. Releasing the ownership guarantees trace buffer will not be accessed.

Once ownership is released, all other resources allocated for the tracer can be released as well.

27.4 Diagnostic Counters

The device supports a mechanism, which allows periodic sampling of diagnostic “counters”.

SW configures a set of counters and the “*sampling period*”. Every sampling period, the device stores a “*sample*” which includes all sampled counters and sampling time stamp.

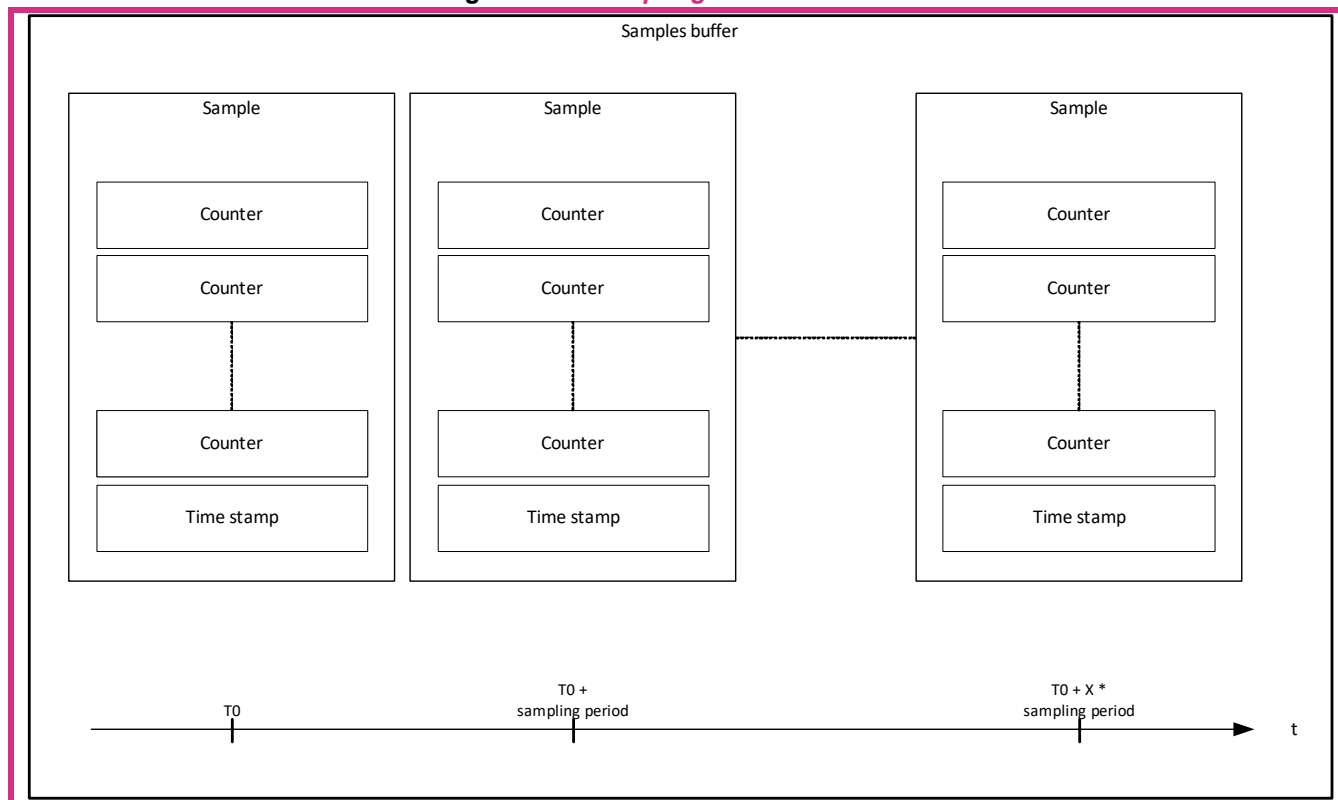
SW configures the number of samples to be stored in the “*samples buffer*”. The host can fetch these samples in a single query.

Diagnostic Counters supported when HCA_CAP.debug==1, and enabled by Diagnostic Counters Commands [See TBD](#).

The device supports two working methods:

- Single sampling – the samples are stored and once the storage has been filled, sampling is terminated.
- Repetitive sampling – when the sample buffer is filled, new samples override old samples.

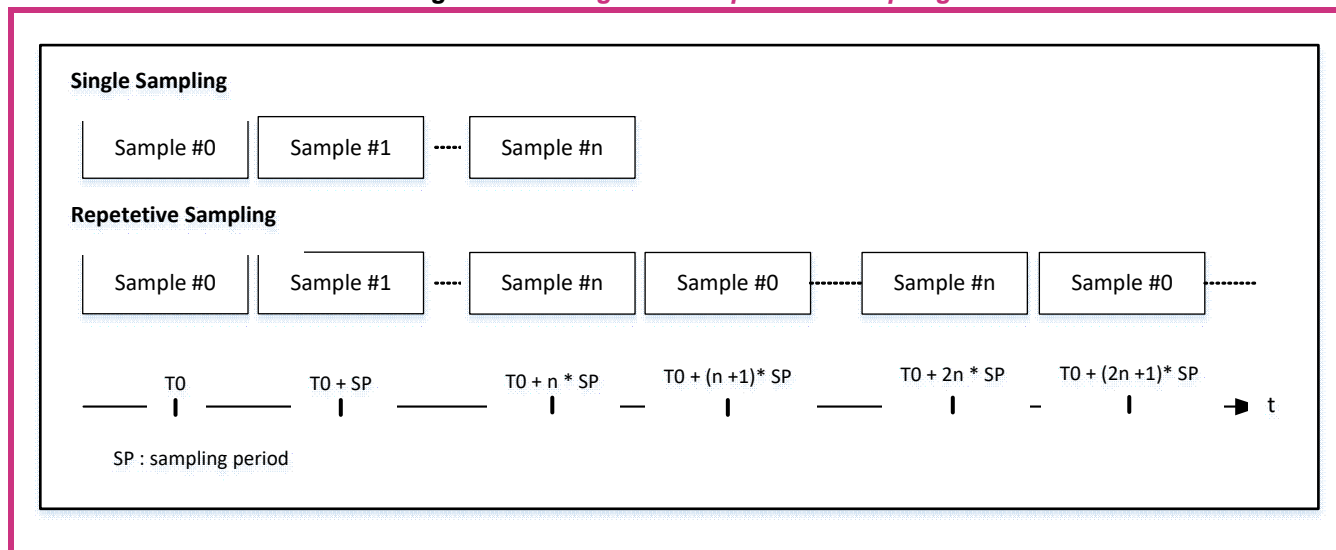
Figure 111: Sampling Buffer Overview



The device expose two types of sampling mechanisms:

- **Diagnostics Counters** which are sampled by the device every period. These counters may have some positive clock drift.
- **Synced Start Diagnostics Counters** which have the ability to count for the measuring period i.e. the trigger for all counters is synced. Synced Start Diagnostic Counters have the optional ability to be cleared at the beginning of each period.

Figure 112: Single and Repetitive Sampling



Note: The `set_diagnostics` and `query_diagnostics` commands are accessible only for port owner.

Note: Once a function has set the `number_of_samples` to be other than 0, it has become a diag ownership. Only this function can access the `set_diagnostics` and `query_diagnostics` commands.

In order to release diag ownership, the owner shall set `number_of_samples` to 0.

27.5 Health Monitoring

27.5.1 RX Path Activity

In the event where the device unexpectedly becomes unresponsive for a long period of time, flow control mechanism may propagate pause frames which will cause congestion spreading to the entire network. To prevent this scenario, the device monitors its status continuously, attempting to detect when it becomes stalled. When the device is stalled for a period longer than a pre-configured timeout, flow control mechanisms (Global Pause and PFC) are automatically disabled. When PFC is in use, and one or more priority has become stalled, PFC will be disabled on all priorities. When the device detects that the stall has ceased, flow control mechanism will resume to its configured behavior.

The configuration is per physical port. If more than a single host is configuring this configuration, the last configuration will be used.

The support for device stall detection is indicated by `HCA_CAP.Debug_CAP.health_mon_rx_activity` (See Section 29.3.4.10, “Debug Capabilities,” on page 1777)

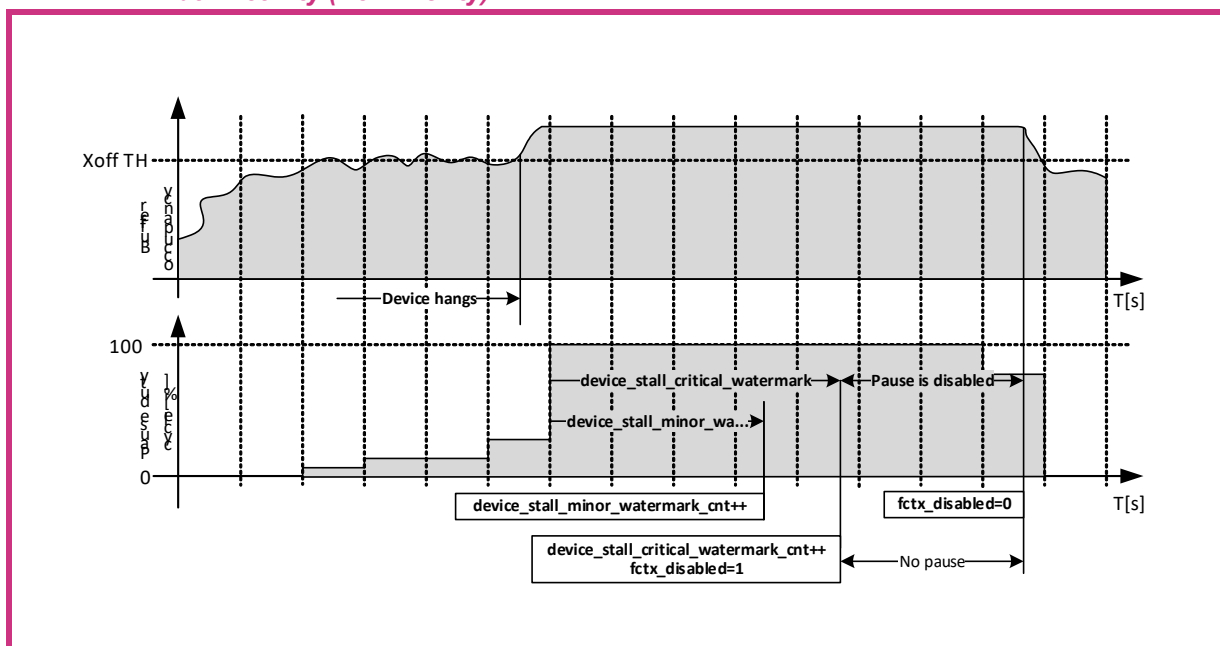
The device supports two thresholds:

- **device_stall_minor_watermark (PFCC)** – when the device is stalled for a period longer than a pre-configured watermark, a diagnostic counter is incremented, allowing the debug utility an insight into current device status. The watermark is set in

device_stall_minor_watermark and the number of times this threshold was crossed is indicated in the device_stall_minor_watermark_cnt (PPCNT)

- device_stall_critical_watermark (PFCC) - when the device is stalled for a period longer than a pre-configured timeout, the pause transmission is disabled, the fctx_disabled (PFCC) is set and the device_stall_critical_watermark_cnt (PPCNT) is advanced

Rx Path Activity (Per Priority)



27.6 Device Health Events

The device deploys self-monitoring mechanisms for detecting health issues. Health issues that do not totally hang the device (e.g it may still execute SW commands) are reported by a General Notification event (subtypes 0x2 and 0x3).

27.7 PCIe Hanged Event

Note: This feature is supported when `HCA_CAP.pcie_hanged==1`.

This feature senses the host's PCIe activity. The sense is done according to PCIe credits activity. In case credits are not released for a time longer than configured, the hang mechanism will be activated, and the packet with the hanged host as destination will be dropped, releasing occupied buffers.

Once the host releases the credits, traffic will resume automatically.

The credit release time is configured through `PCIe_credit_token_TO` parameter. Note that the `PCIe_credit_token_TO` sets the sampling period for PCIe hang. The sensing time might take up to two times the size of the sampling period.

By default, the timeout is configured as 'auto'. In this mode, the TO is configured according to PCIe configuration space "Completion Timeout Value", where the `NV_GLOBAL_PCI_CONFIG_2.PCIe_credit_token_TO` is the maximum "Completion Timeout Value".

In case hanging takes place, all other hosts will get 'PCIe_hang_event'. See [Table 215, "General Notification Event Subtype," on page 496](#). Note that this feature is disabled when PCIe "Completion Timeout Disable" is set.

27.8 Resource Dump

`resource_dump` command is used for debugging and analyses purposes of the device activities. Using this command, resources are dumped from the device to the user.

`resource_dump` uses segment structures. The segment type allows the user to select the desired dump. It is also used for structuring the dump response. Segments are described in [Section 27.8.1, "Segments," on page 1062](#).

Description on how to use the `resource_dump` command can be found in [Section 27.8.2, "Resource Dump Command Usage," on page 1075](#).

The device may generate events notifying resource dumps updates. Segments that support events generation provide this capability in the menu segment in the appropriate record. The event is described in `resource_dump_event` (see [Section 8.20.11.3, "Resource Dump Event," on page 489](#)).

The event contains the needed information to call the `resource_dump` command

Event sequence

Ensure that the eq to be used for the events is at least `debug_capabilities.log_min_resource_eq`

Arm required segments using the `ARM_RESOURCE_DUMP_EVENT` command for `segment_type` field to be equal to menu segment type.

Once event has occurred, user might call `ARM_RESOURCE_DUMP_EVENT` command with the information provided in the event. The device will dump the resource in the standard resource dump sequence and the related resource dump will be cleared.

In order to re-arm the event, user shall call `resource_dump` with the same information which was provided in the event...

27.8.1 Segments

Segments are used as a container for `resource_dump` command response. The device can return one or more segments for a single `resource_dump` request.

Segments are divided into two categories:

- **Control segments** - provide pre-defined and pre-structured information. The control segments are:
 - Reference segment - points to a sub-segment (a leaf) of the current segment
 - Menu segment and info segment - contain device capabilities and information
 - Command segment - identifies and isolates multiple resource dump commands
 - Notice segment - provides notice on the dump

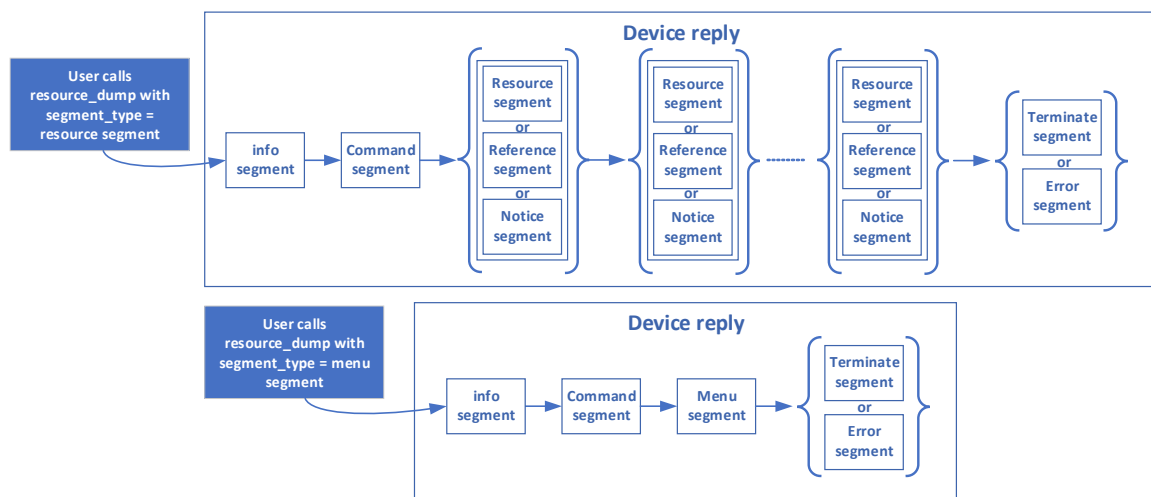
- Terminate and error segments - act as the last segments in a dump

Control segment ranges between segment_type 0xff00 to 0xffff

- **Resource segments** - provide device specific dump resource
 - The internal content of the Resource segment is out of the scope of this document
 - Resource segment must always be accompanied with control segments
 - Resource segment ranges between segment_type 0x0000 to 0xfeff

Figure 113 below describes the segment list format as a result of the resource_dump command.

Figure 113: Segment Format



Each segment is comprised of a segment header and a segment-specific structure.

The following sections provide information on the control and resource segments format.

27.8.1.1 Segment Header

Table 823 - Segment Header Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
length_dw																segment_type												00h				

Table 824 - Segment Header Format

Offset	Bits	Name	Description
00h	31:16	length_dw	The segment length from 00h till the end of segment in dwords units
	15:0	segment_type	The type of the segment, range of 0x0000 to 0xffef

27.8.1.2 Segment-Specific Structure

27.8.1.2.1 Resource Segment

This segment contains the dump results of a certain resource.

Segment properties:

- The segment size depends on the type of the segment dump
- Returned as a part of the output for Resource dump
- Multiple resource segments can appear in the segment list

Table 825 - Resource Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823 , “Segment Header Layout,” on page 1064)																															00h	
																																04h
index1																															08h	
index2																															0Ch	
payload[0]																															10h	
payload[1]																															14h	
...																															...	

Table 826 - Resource Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823 , “Segment Header Lay- out,” on page 1064)	Segment range is from segment_type 0x0000 to 0xfeff
08h	31:0	index1	First index of dumped segment (optional, 0 if not used)
0Ch	31:0	index2	Second index of dumped segment (optional, 0 if not used)
10h-...	31:0	payload[...]	Segment dump

27.8.1.2.1.1Pre-defined Resource Segments

The following is a list of segments that are pre-defined.

Table 827 - Pre-defined Resource Segments

Segment_name	Segment_type	Index1_name	Support_index1	Must_have_index1	Index2_name	Support_index2	Must_have_index2	Support_num_of_obj1	Must_have_num_of_obj1	num_of_obj1_supports_all	num_of_obj1_supports_active	Support_num_of_obj2	Must_have_num_of_obj2	num_of_obj2_supports_all	num_of_obj2_supports_active	Event_supported
MLX5_S-GMT_TYPE_SX_SLICE_ALL	0x3001		No	No		No	No	No	No	No	No	No	No	No	No	No
MLX5_S-GMT_TYPE_RX_SLICE_ALL	0x2001		No	No		No	No	No	No	No	No	No	No	No	No	No
MLX5_S-GMT_TYPE_FULL_QPC	0x1000	QP number	Yes	Yes		No	No	No	No	No	No	No	No	No	No	No
MLX5_S-GMT_TYPE_SND_BUFF	0x1001	QP number	Yes	Yes	WQ EBB	Yes	No	No	No	No	No	Yes	No	Yes	No	No
MLX5_S-GMT_TYPE_RCV_BUFF	0x1002	QP number	Yes	Yes	DS	Yes	No	No	No	No	No	Yes	No	Yes	No	No
MLX5_S-GMT_TYPE_GVMI_SYND_REC	0x1010		No	No		No	No	No	No	No	No	No	No	No	No	Yes
MENU	0xffff		No	No		No	No	No	No	No	No	No	No	No	No	No

Table 827 - Pre-defined Resource Segments

Segment_name	Segment_type	Index1_name	Support_index1	Must_have_index1	Index2_name	Support_index2	Must_have_index2	Support_num_of_obj1	Must_have_num_of_obj1	num_of_obj1_supports_all	num_of_obj1_supports_active	Support_num_of_obj2	Must_have_num_of_obj2	num_of_obj2_supports_all	num_of_obj2_supports_active	Event_supported
5_SGMT_TYPE_QUE- RY_CQ	0x4020	CQN	Yes	Yes	N/A	No	No	No	No	No	No	No	No	No	No	No
MLX5_S- GMT_ - TYPE_QUE- RY_MKEY	0x4030	MKEY_IDX	Yes	Yes	N/A	No	No	No	No	No	No	No	No	No	No	No
MLX5_S- GMT_ - TYPE_QUE- RY_QP	0x4000	QPN	Yes	Yes	N/A	No	No	No	No	No	No	No	No	No	No	No

27.8.1.2.2 Notice Segment

This segment contains a textual notice for the user, such as “issue occurred during dump”

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump
- May appear anywhere in the segments list
- Multiple notice segment can appear in the segment list

Table 828 - Notice Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																																00h	
																syndrome_id																04h	
																																	08h-0Ch
notice																																10h-2Ch	

Table 829 - Notice Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823 , “Segment Header Lay- out,” on page 1064)	segment_type is 0xfff9
04h	15:0	syndrome_id	Device syndrome for the notice (optional, 0 if not used)
10h-2Ch	256	notice	A string containing the notice (32 bytes) (info_string)

27.8.1.2.3 Command Segment

This segment contains resource_dump call information. It is added to the output to allow easy parsing.

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump
- Must be sent as the second segment in the segment list

Table 830 - Command Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																															00h	
segment_called															vhca_id																04h	
index1																															08h	
index2																															0Ch	
num of obj1															num of obj2																10h	

Table 831 - Command Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment_type is 0xffff
04h	31:16	segment_called	The segment required by the resource_dump command
	15:0	vhca_id	The vhca_id required by the resource_dump command
08h	31:0	index1	First index of required by the resource_dump command
0Ch	31:0	index2	Second index of required by the resource_dump command
10h	31:16	num of obj1	First num_of_obj required by the resource_dump command
	15:0	num of obj2	Second num_of_obj required by the resource_dump command

27.8.1.2.4 Terminate Segment

Indicates that the dump was terminated with no error on the device’s end.

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump or menu dump

Must be sent as the last segment in the segment list

Table 832 - Terminate Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																															00h	

Table 833 - Terminate Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment_type is 0xffffb

27.8.1.2.5 Error Segment

Indicates that the dump was terminated with an error on the device’s end.

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump or menu dump
- Must be sent as the last segment in the segment list

Table 834 - Error Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																															00h	
																syndrome_id															04h	
																																08h-0Ch
error																															10h-2Ch	

Table 835 - Error Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment_type is 0xfffc
04h	15:0	syndrome_id	Device syndrome for the error (optional, 0 if not used)
10h-2Ch	256	error	A string containing the error (32 bytes) (info_string)

27.8.1.2.6 Reference Segment

This segment provides information on a reference segment that is a sub segment of the current segment dump.

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump
- Multiple reference segments can appear in the segment list

Table 836 - Reference Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																															00h	
																reference_segment															04h	
index1																															08h	
index2																															0Ch	
num of obj1																num of obj2															10h	

Table 837 - Reference Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment_type is 0xffffd
04h	15:0	reference_seg- ment	The number of the referred segment
08h	31:0	index1	First index of the referred segment
0Ch	31:0	index2	Second index of the referred segment
10h	31:16	num of obj1	First num_of_obj of the referred segment
	15:0	num of obj2	Second num_of_obj required of the referred segment

27.8.1.2.7Info Segment

A segment that provides Resource information on the device.

Segment properties:

- The segment length is fixed
- Returned as a part of the output for Resource resource dump or menu dump

Must be sent as 1st segment in the segment list

Table 838 - Info Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																															00h		
																																dump_version	04h
hw_version																															08h		
fw_version																															0Ch		

Table 839 - Info Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment_type is 0xfffe
04h	7:0	dump_version	Dump format version (currently, 0 is used)
08h	31:0	hw_version	HW version for parsing
0Ch	31:0	fw_version	FW version for parsing

27.8.1.2.8 Menu Segment

This segment provides a list of all Resource segments supported by the device.

Segment properties:

- The segment size depends on number of Resource segments supported
- Returned as a part of the output for menu dump

Table 840 - Menu Segment Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
segment_header (See Table 823, “Segment Header Layout,” on page 1064)																																00h
																num_of_records																04h
menu_record[0](See Table 842, “Menu Record Layout,” on page 1074)																																08h-38h
menu_record[1](See Table 842, “Menu Record Layout,” on page 1074)																																3Ch-6Ch
...																																...

Table 841 - Menu Segment Format

Offset	Bits	Name	Description
00h	31:0	segment_header (See Table 823, “Segment Header Layout,” on page 1064)	segment type is 0xffff
04h	15:0	num_of_records	The number of Resource segments records in the list
08h-...	416	menu_record[...](See Table 842, “Menu Record Layout,” on page 1074)	See Table 842, “Menu Record Layout,” on page 1074 .

Table 842 - Menu Record Layout

Offset	00h	04h-10h	14h-20h	24h-30h		
31	segment_type	segment_name			index1_name	index2_name
30						
29						
28						
27		num_of_obj2_supports_active	event supported			
26		num_of_obj2_supports_all				
25		must_have_num_of_obj2				
24		support_num_of_obj2				
23		num_of_obj1_supports_active				
22		num_of_obj1_supports_all				
21		must_have_num_of_obj1				
20		support_num_of_obj1				
19		must_have_index2				
18		support_index2				
17		must_have_index1				
16		support_index1				
15						
14						
13						
12						
11						
10						
9						
8						
7						
6						
5						
4						
3						
2						
1						
0						

Table 843 - Menu Record Format

Offset	Bits	Name	Description
00h	28	event_supported	When set, the segment supports segment update event reporting
	27	num_of_obj2_-supports_active	
	26	num_of_obj2_-supports_all	
	25	must_have_num_of_obj2	
	24	support_num_of_obj2	
	23	num_of_obj1_-supports_active	
	22	num_of_obj1_-supports_all	
	21	must_have_num_of_obj1	
00h	20	support_num_of_obj1	
	19	must_have_index2	
	18	support_index2	
	17	must_have_index1	
	16	support_index1	
	15:0	segment_type	
04h-10h	128	segment_name	
14h-20h	128	index1_name	
24h-30h	128	index2_name	

27.8.2 Resource Dump Command Usage

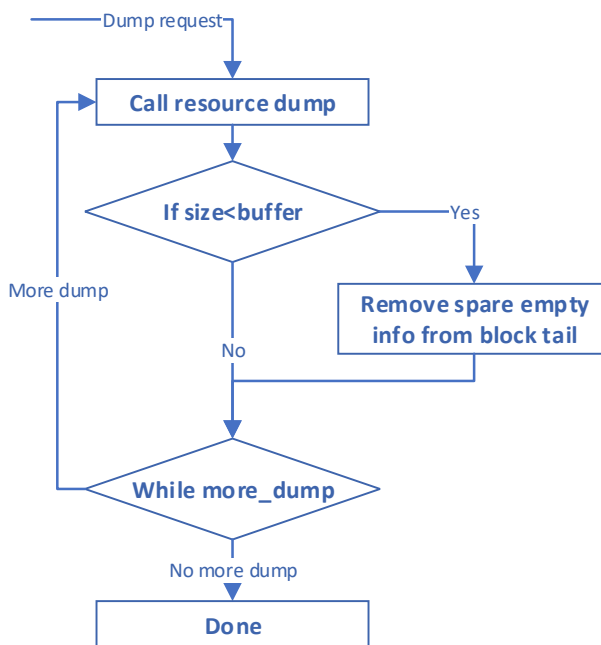
resource_dump command is used to perform the dump. The response is divided into blocks, and the user shall concatenate the block into one buffer.

The command can retrieve the dump using one of the following modes:

- **Inline mode** - this mode is selected by setting `inline_dump` bit to '1'. When using inline mode, response is returned over relatively small blocks in the command structure (`inline_Resource` field). When working in inline mode, all `mkey` fields (`mkey`, `address` and `size`) must be set to zero.
- **Mkey mode** - block is according to the size allocated in `mkey`. Mkey has the following requirements:
 - `Mkey.start_address` must be aligned to 4K
 - `Mkey.length` must be aligned to 4K
 - `Input.address` must be aligned to DWORD
 - `Input.size` must be aligned to DWORD

27.8.2.1 Resource Dump Sequence

Figure 114: Resource Dump Sequence



27.8.2.1.1 Menu Usage

Menu segment uses the same mechanism. Menu segment contains the segment supported by the device and the expected input for each segment.

User shall verify the compliance of the desired segment with the menu segment before calling first resource dump.

27.8.2.1.2 Call Sequence

27.8.2.1.2.1 First Calls

Resource Dump is initiated with the user providing:

- `segment_type`, `index1`, `index2`, `num_of_obj1`, `num_of_obj2`
 - The command shall be called with minimum params as defined in menu segment for this `segment_type`
 - The command shall not contain unsupported params as defined in menu segment for this `segment_type`
- `vhca_id`
- `inline_dump` bit and mkey resource (`mkey`, size address) if mkey is used
- `device_opaque` & `more_dump` must be set to 0

Upon command return:

- Resource dump is returned in the allocated mkey memory or inline, depending on the selected mode
- Size indicates the number of valid bytes that were returned in the mkey allocated memory or inline
- `more_dump` indicated whether additional call to `resource_dump` is required to get the rest of the dump. In case `more_dump == 1`, proceeding calls are needed
 - Device response might indicate that `more_dump` is needed even if the returned size is smaller than the allocated buffer. In that case, the unused bytes after size shall be ignored

27.8.2.1.2.2 Proceeding Calls

In case the `more_dump` was set in the device reply, user shall recall the `resource_dump` command `resource_type`, with `index1`, `index2`, `num_of_obj1`, `num_of_obj2`, `device_opaque` and `more_dump` set to the same value that was returned by the device upon last return. `segment_type` shall be kept as was in the original call.

`Seq_num` field is incremented for each call made after the first call. It can be used to verify that no return command was lost.

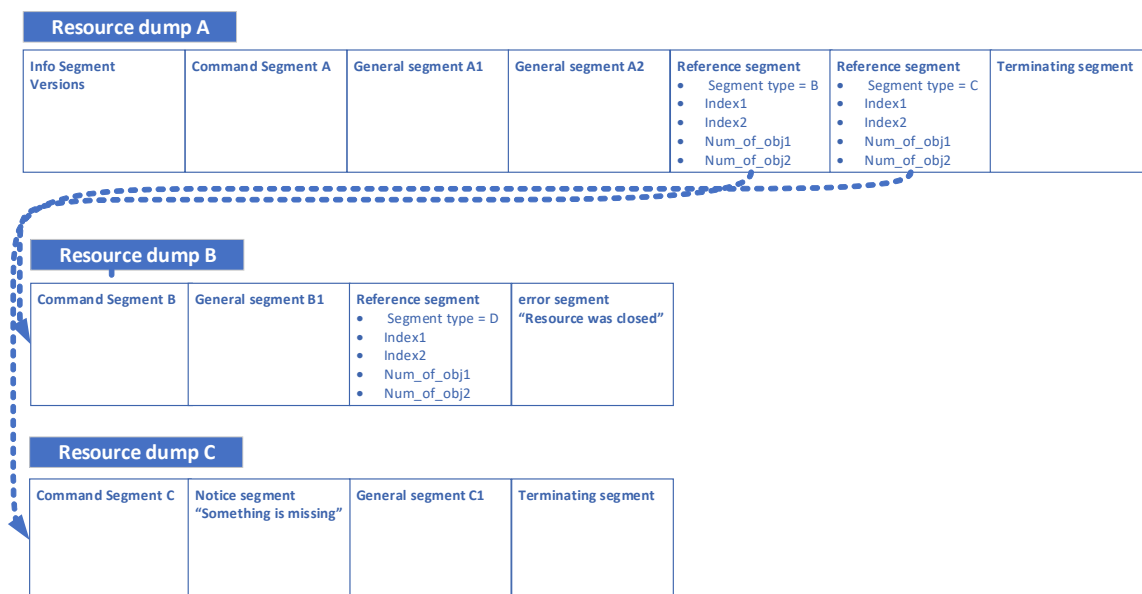
In case `more_dump` was set in the device reply, user shall recall the `resource_dump` command. `resource_type`, with `seq_num`, `more_dump`, `index1`, `index2`, `num_of_obj1`, `num_of_obj2`, `device_opaque` and `more_dump` are set to the same value that was returned by the device upon last return. `segment_type` shall be kept as was in the original call.

27.8.2.1.3 Reference Usage

Reference segments are used to allow the user to dump leaves of the resource root. For example, a QP dump might carry reference segments to its CQs.

The user can call the leaf by using the content of reference segment to call resource dump

Figure 115: 2-Tier Dump Example



27.9 Core Dump

The CORE_DUMP register (See Section 28.11, "CORE DUMP Register," on page 1128) may be used by SW to instruct the device to dump internal debug information to its trace buffer. The driver may invoke CORE_DUMP either upon encountering an error or upon user demand.

The CORE_DUMP register is a Write-Only register that instructs the device to dump internal debug information to the device tracer buffer. The "core_dump_type" field of the register tells the device what type of device information is requested. The information written by the device to its trace buffer is device-internal and thus it is not described in the PRM. The command completion does not depend on the device dump completion, it rather only launches the device information dumping task.

27.10 Stress Generation

Hca testing depends on running a variety of stimuli in order to reach corner cases. FW stress hooks utilizes both HW and FW DFT (design for testing) in order to increase stress. Corner cases can stem from one of following:

- Traffic stress – especially important for HW related bugs. Multiple parallel packet handling / FW exception flows are required.
- Unlikely branches – common flows can often have corner cases which involve different FW control (for instance: no pages during command, CQE GW busy, EQE GW busy)
- Code hot spots – some parts of FW code are risky, the flows to activate them can sometimes be cumbersome – but the flows themselves should be harmless.

27.10.1 Stress Types

The following stress types are defined:

2. Stop toggler – handles traffic stress by stopping central HW/FW machines for a configurable amount of time. Will ensure machine is stopped/working for T1 time out of every T2 duty cycle. See “STRS_STOP_TOGGLE Register”
3. Fault injector – reaches unlikely branches by injecting a busy/fail fault N times every M function calls. See “STRS_FAULT_INJECT Register”
4. Mini flow activator – hits HW/FW hot spots by calling internal FW functions (background noise) once every T time. See “STRS_MINI_FLOW Register”
5. Resource reduction – reduces HW / FW resources, creates artificial bottlenecks. See “STRS_RESOURCE Register”

Multiple hooks of each kind can be activated simultaneously.

The support for Stress Generation is indicated by str_cap field in ICMD_QUERY_CAPABILITY.

28 Command Registers

28.1 Access Reg Space

The chip's access register form are spread out in a single address domain. Each register has it's own address and length.

Table 844 describes the register space and there various properties.

The various columns are described in Table 844 below:

Table 844 - Access Register Table Columns:

Name	Values
write type	(V) - volatile - register will revert to default value on any reset (R) - non-volatile for PCI reset, however, register will lose its value on full power cycle. (P) - non volatile - register value will retain its value over full power cycle
length type	(F) - full - register only supports writes for entire register length (P) - partial - register supports any length (up to maximum register size)

28.2 Access Registers

Table 845 summarizes the access register set.

Table 845 - Access Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Init/ runtime	Reference
PORT_INFO_RO_FIELDS	0x5020					See Table 846, "PORT_INFO_RO_FIELDS Parameters Layout," on page 1083
NODE_INFO_RO_FIELDS	0x5021					See Table 848, "NODE_INFO_RO_FIELDS Parameters Layout," on page 1084
UNIT_PERF_COUNTERS_GLOBAL_CTRL	0x100	v	v		R	See Table 883, "UNIT_PERF_COUNTERS_GLOBAL_CTRL Register Layout," on page 1098
UNIT_PERF_COUNTERS_CFG		v	v		R	See Table 885, "UNIT_PERF_COUNTERS_CFG Register Layout," on page 1099
UNIT_PERF_COUNTERS_DATA		v	v		R	See Table 887, "UNIT_PERF_COUNTERS_DATA Register layout," on page 1101
MISC_COUNTERS	0x104	v	v		R	See Table 889, "MISC_COUNTERS Register Layout," on page 1102
Loopback_Control_Register	0x0700	v	v		I	See Table 874, "Loopback Control Register," on page 1095

Table 845 - Access Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Init/ runtime	Reference
DB_TRESHOLD	0x0701	v	v	V	R	See Table 878, “DB_THRESHOLD Register Layout,” on page 1096
DCBX_PARAM	0x4020	v	v			See Table 860, “DCBX_PARAM Register Layout,” on page 1087
DCBX_APPLICATION	0x4021	v				See Table 864, “DCBX_APPLICATION Register Layout,” on page 1090
FPGA_CAP	0x4022	v				See Table 891, “FPGA_CAP Register Layout,” on page 1103
FPGA_CTRL	0x4023	v	v			See Table 895, “FPGA_CTRL Register Layout,” on page 1108
FPGA_ACCESS_REG_SPACE	0x4024	v	v			See Table 897, “FPGA_ACCESS_REG_SPACE Register Layout,” on page 1111
FPGA_SHELL_COUNTERS	0x4025	v	v			See Table 899, “FPGA_SHELL_COUNTERS Register Layout,” on page 1112
Quality of Service Registers	0x4000-0x4fff (except of 0x4020-0x402f)	v	v			See Table 710, “QoS Register Summary,” on page 930
STRS_CAP	0x4026	v				See Table 901, “STRS_CAP Register Layout,” on page 1113
STRS_STOP_TOGGLE	0x4027	v	v			See Table 903, “STRS_STOP_TOGGLE Register Layout,” on page 1114
STRS_FAULT_INJECT	0x4028	v	v			See Table 921, “STRS_FAULT_INJECT Register Layout,” on page 1122
STRS_MINI_FLOW	0x4029	v	v			See Table 925, “STRS_MINI_FLOW Register Layout,” on page 1124
STRS_RESOURCE	0x402A	v	v			See Table 931, “STRS_RESOURCE Register Layout,” on page 1127
ROCE_ACCL	0x402C	v	v			See Table 868, “ROCE_ACCL Register Layout,” on page 1092
SECURE_HOST	0x402D	v	v			See Table 933, “SECURE_HOST Register Layout,” on page 1127
CORE_DUMP	0x402E		v			See Table 935, “CORE_DUMP Register Layout,” on page 1128

Table 845 - Access Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Init/ runtime	Reference
CAPI_CTRL	0x402f	v	v			See Table 939, “CAPI_CTRL Register Layout,” on page 1132
Ports_Control_Registers	0x5000-0x5fff					See Table 951, “Ports Register Summary,” on page 1140
SET_NODE	0x6001					See Table 850, “SET_NODE Input Structure Layout,” on page 1084
POWER_SETTINGS	0x7001					See Table 852, “POWER_SETTINGS Register Layout,” on page 1085
FP_SL_MAP	0x7002					See Table 854, “FP_SL_MAP - Argument Layout,” on page 1085
HOST_ENDIANNESS	0x7004	v	v			See Table 858, “HOST_ENDIANNESS Register Layout,” on page 1086
Management and Peripherals_control_registers	0x9000-0x9fff	v	v			See Table 1326, “Management and Peripherals Control Register Summary,” on page 1492
Buffer_management_registers	0xB000-0xBFF	v	v			See Table 1233, “Congestion Control Register Summary,” on page 1421
Congestion_control_registers	0x2000-0x2FFF	v	v			See Table 1268, “Shared Buffer Management Register Summary,” on page 1450
NIC_only_registers	0xC000-0xCFFF	v	v			
Resource_dump_registers	0xC000	v	v			See Table 937, “RESOURCE_DUMP Register Layout,” on page 1130
NCFG	0xC001	v	v			See Table 941, “NCFG Register Layout,” on page 1134
CRYPTO_OPERATIONAL	0xC002	v	v			See Table 943, “CRYPTO_OPERATIONAL Register Layout,” on page 1135
CRYPTO_COMMISSIONING	0xC003	v	v			See Table 945, “CRYPTO_COMMISSIONING Register Layout,” on page 1136
IMPORT_KEK_HANDLE	0xC004	v	v			See Table 947, “IMPORT_KEK_HANDLE Register Layout,” on page 1137
CREDENTIAL_HANDLE	0xC005	v	v			See Table 949, “CREDENTIAL_HANDLE Register Layout,” on page 1138

28.2.1 PORT_INFO_RO_FIELDS

PORT_INFO_RO_FIELDS register can be used to modify PortInfo parameters that are read only to MADs, for example, *capability_mask*. Those parameters can be set through the ModifyHCAAttributes verb. SET_PORT is also used to set up capabilities lower than the maximum allowed by QUERY_HCA_CAP.

Note that reading the PortInfo can be done using the MAD_IFC command. For changing the *capability_mask*, it is recommended to read the current port capability mask by submitting a Get-PortInfo to the MAD_IFC command. The new mask should then be written using this register.

This register can be accessed regardless of whether the port is already up. It is recommended that certain parameters such as GID table size, are changed while the port is inactive in order to avoid atomicity issues.

Table 846 - PORT_INFO_RO_FIELDS Parameters Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								port								max_gid																00h
																																04h
port_guid																																08h-0Ch

Table 847 - PORT_INFO_RO_FIELDS Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:16	port		
	15:0	max_gid	Maximum number of GIDs for the port.	
08h-0Ch	64	port_guid	Port GUID	

28.2.2 NODE_INFO_RO_FIELDS

NODE_INFO_RO_FIELDS register can be used to modify NodeInfo parameters that are read only to MADs. Those parameters can be set through the ModifyHCAAttributes verb. It is also used to set up capabilities lower than the maximum allowed by QUERY_HCA_CAP.

Note that reading the NodeInfo can be done using the MAD_IFC command.

This register can be accessed regardless of whether the port is already up. It is recommended that certain parameters such as PKey table size, are changed while the port is inactive in order to avoid atomicity issues.

Table 848 - NODE_INFO_RO_FIELDS Parameters Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
system_image_guid																															00h-04h	
																															08h-0Ch	
node_guid																															10h-14h	
															max_pkey																18h	
																															1Ch	

Table 849 - NODE_INFO_RO_FIELDS Parameters Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	system_image_guid	System Image GUID, takes effect only if the sig bit is set.	
08h-0Ch	64	guid0	EUI-64 GUID assigned by the manufacturer.	
10h-14h	64	node_guid	Node GUID,	
18h	15:0	max_pkey	Maximum PKeys for the port.	
1Ch	31:0	reserved		

28.2.3 SET_NODE - Set Node Information

Table 850 - SET_NODE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
node_description																																0h-3Ch

Table 851 - SET_NODE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0h-3Ch	512	node_description	Text describing the node	

28.3 MISC Registers

28.3.1 POWER_SETTINGS Register

The POWER_SETTINGS register controls array power save feature. When power save is activated, arrays will power down when the chip is idle. This will result in increased latency when chip leaves idle state.

Table 852 - POWER_SETTINGS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								power_settings_level				00h				
																																04h
																																08h
																																0Ch

Table 853 - POWER_SETTINGS Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	power_settings_level	0x0 - minimum power (maximum latency penalty) 0x1-0xfe - reserved 0xff - maximum power (minimum latency penalty)	RW

28.3.2 FP_SL_MAP Register

The FP_SL_MAP register configures and reports the mapping of SLs to fast path. This can be accessed per port (specified the *argument* field in the ACCESS_REGISTER command).

Table 854 - FP_SL_MAP - Argument Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								port				00h				

Table 855 - FP_SL_MAP - Argument Field Descriptions

Offset	Bits	Name	Description	Access
00h	7:0	port	0x0 - reserved 0x1-0x2 - port number 0x3-0xff - reserved	

Table 856 - FP_SL_MAP - Fast Path VL Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																fp_sl_indication																00h
																																04h
																																08h
																																0Ch

Table 857 - FP_SL_MAP - Fast Path VL Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	15:0	fp_sl_indication	If bit [i] of the register is set, SL[i] is a fast path SL	RW

28.3.3 HOST_ENDIANNESS Register

The HOST_ENDIANNESS register configures endianness for atomic operations.

Table 858 - HOST_ENDIANNESS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
he																															00h	
																															04h	
																															08h	
																															0Ch	

Table 859 - HOST_ENDIANNESS Register Fields Description

Offset	Bits	Name	Description	Access
00h	31	he	0: LITTLE_ENDIAN 1: BIG_ENDIAN	RW

28.3.4 DCBX_PARAM Register

This register queries and configures DCBX capabilities.

Table 860 - DCBX_PARAM Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
debx_cce_cap	debx_ieee_cap	debx_standby_cap						port_number															max_application_table_size									
																				version_oper						version_admin						
willing_admin			pfc_cap_oper						pfc_cap_admin						num_of_tc_oper						num_of_tc_admin											
remote_willing			remote_pfc_cap															remote_num_of_tc														
																							error									

Table 861 - DCBX_PARAM Register Fields

Offset	Bits	Name	Description	Access
00h	31	dcbx_cee_cap	If set, DCBX CEE version is supported.	RO
	30	dcbx_ieee_cap	If set, DCBX IEEE version is supported.	RO
	29	dcbx_standby_cap	If set, device supports DCBX on standby mode (aux power).	RO
	23:16	port_number	Physical port number. <i>This field is reserved when HCA_CAP.num_ports == 1. See Table 1567, "QUERY_HCA_CAP Input Structure Layout," on page 1667</i>	Index
	5:0	max_application_table_size	The maximum size of DCBX application table.	RO
04h	10:8	version_oper	0x0: disabled - host is in charge. 0x1: CEE_enabled 0x2: IEEE_enabled If the DCBX_PARAM.version_oper is 0x0, all oper and remote fields in DCBX_PARAM () are invalid. DCBX_APPLICATION.num_app_prio is 0 DCBX_PARAM.version is 0x0 until a successful handshake with remote party. <i>[internal]: 1 & 2 are set only after negotiation with remote party.</i>	RO
	2:0	version_admin	Set the DCBX version. 0x0: disabled- host is in charge 0x1: CEE - supported only when dcbx_cee_cap==1 0x2: IEEE - supported only when dcbx_ieee_cap==1 0x3: CEE_and_IEEE - CEE and IEEE auto select. When using this mode, the device uses IEEE mode. If handshake with other party succeed, it will continue on using IEEE, otherwise it will fall back to CEE. Supported only when dcbx_cee_cap==1 and cbx_ieee_cap==1.	RW

Table 861 - DCBX_PARAM Register Fields

Offset	Bits	Name	Description	Access
08h	31	willing_admin	0x0: non_willing 0x1: willing	RW
	27:24	pfc_cap_oper	Number of traffic classes that may simultaneously support PFC. Can be set in the range 0...8 - operational state after negotiation with remote party. <i>[intenal]: In case of link down or reset of fw, the information in this register shall be reset></i>	RO
	19:16	pfc_cap_admin	Number of traffic classes that may simultaneously support PFC. Can be set in the range 0...8	RW
	11:8	num_of_tc_oper	Number of TCs - operational state after negotiation with remote party. <i>[intenal]: In case of link down or reset of fw, the information in this register shall be reset></i>	RO
	3:0	num_of_tc_admin	Number of TCs.	RW
0Ch	31	remote_willing	Remote side's willing state. Valid only after negotiation with remote party	RO
	27:24	remote_pfc_cap	Remote side's number of traffic classes that may simultaneously support PFC. Valid only after negotiation with remote party.	RO
	3:0	remote_num_of_tc	Remote side's number of TCs. Valid only after negotiation with remote party.	RO
10h	7:0	error	Bitmask indicates errors reported. Bit 0: not enough TC to comply with remote Bit 1: not enough pfc to comply with remote Bit 2: TSA unsupported	RO

28.3.5 MMHI - Multi-Host Information Register

This register provides a privileged host information about other hosts in its system.

Table 862 - MMHI Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																host_en											00h					
																								host_number					04h			
																																08h-7Ch

Table 863 - MMHI Register Field Description

Offset	Bits	Name	Description	Access
00h	15:0	host_en	Bit per Host indicating this host is valid in the current Multi-Host configuration	RO
04h	7:0	host_number	Identifier for the host used to access the register	RO

28.3.6 DCBX_APPLICATION Register

This register provides the application priority DCBX's TLVs. Only remote device query is supported.

An update to this table will trigger a DCBX_APPLICATION_PARAM change event.

Table 864 - DCBX_APPLICATION Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
									port_number																								00h
																										num_app_prio			04h				
																												08h-0Ch					
app_prio[0] (See Table 866, “Application Priority Entry Layout,” on page 1091)																												10h					
app_prio[1] (See Table 866, “Application Priority Entry Layout,” on page 1091)																												14h					
...																												18h...					

Table 865 - DCBX_APPLICATION Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	port_number	Physical port number. <i>This field is reserved when HCA_CAP.num_ports == 1. See Table 1567, "QUERY_HCA_CAP Input Structure Layout," on page 1667</i>	Index
04h	5:0	num_app_prio		RO

Table 865 - DCBX_APPLICATION Register Fields

Offset	Bits	Name	Description	Access
10h-...	31:0	app_prio[...] (See Table 866, “Application Priority Entry Layout,” on page 1091)	Application priority entry.	RO

Table 866 - Application Priority Entry Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
												priority			sel		protocol_id															

Table 867 - Application Priority Entry Fields

Offset	Bits	Name	Description	Access
00h	23:21	priority	This field is reserved when HCA_CAP.num_ports == 1. See Table 1567, “QUERY_HCA_CAP Input Structure Layout,” on page 1667. According to 802.1Q application priority TLV	RO
	18:16	sel	According to 802.1Q application priority TLV	RO
	15:0	protocol_id	According to 802.1Q application priority TLV	RO

28.3.7 ROCE_ACCL - ROCE Accelerator

This register allows enabling the following RoCE traffic accelerator features and is only supported when HCA_CAP.roce_accl==1:

- Exponential back on – automatic tuning of the retransmission time according to path length. This reduces retransmission time.
- Transmission window – changing the transmission window to be smaller than default.
- Slow restart – slow restart of transmission.

Table 868 - ROCE_ACCL Register Layout

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00h	roce_adp_retrans_field_select	roce_tx_window_field_select	roce_slow_restart_field_select																													
04h	roce_adp_retrans_en	roce_tx_window_en	roce_slow_restart_en																													
08h-3Ch																																

Table 869 - ROCE_ACCL; Register Field Description

Offset	Bits	Name	Description	Access
00h	2	roce_slow_restart_field_select	In Query, indicates whether roce_slow_restart mode is supported. If set, indicates roce_slow_restart_en field is valid.	RW
	1	roce_tx_window_field_select	In Query, indicates whether roce_tx_window mode is supported. If set, indicates roce_tx_window_en field is valid.	RW
	0	roce_adp_retrans_field_select	In Query, indicates if roce_tx_windo mode is supported. If set, indicates roce_tx_window_en field is valid.	RW

Table 869 - ROCE_ACCL; Register Field Description

Offset	Bits	Name	Description	Access
04h	2	roce_slow_restart_en	If set, the Slow restart functionality is enabled. Valid only when roce_slow_restart_field_select == 1.	RW
	1	roce_tx_window_en	If set, the Transmission window functionality is enabled. Valid only when roce_tx_window_field_select == 1.	RW
	0	roce_adp_retrans_en	If set, the adaptive retransmission window functionality is enabled. Valid only when roce_adp_retrans_field_select == 1.	RW

28.4 DC Internal Control Register

The DC_INTERANL_CFG register configures DC transport internal configurations.

Table 870 - DC_INTERNAL_CFG - DC Internal Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dcr_garbage_collection_time																															00h	
																															04h	
																															08h	
																															0Ch	

Table 871 - DC_INTERNAL_CFG - DC Internal Configuration Register Fields

Offset	Bits	Name	Description	R/W/ Index
00h	31:0	dcr_garbage_collection_time	Connect-IB FW will check for stale DCR resource every cycle of those period	RW
04h	31:0	reserved		
08h	31:0	reserved		
0Ch	31:0	reserved		

28.5 Schedule Queue Internal Control Register

The SQ_INTERNAL_CFG register configure the size of the schedule queues of the appropriate ports and vls.

Table 872 - SQ_INTERNAL_CFG - Schedule Queue Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sq_base_icm_access																																00h-04h
reserved						num_sq						reserved										log2_stride_size						08h				
reserved																																0Ch
reserved						sq_number_0						reserved						po	rt	VL		log2_sq_size						10h				
reserved						sq_number_1						reserved						po	rt	VL		log2_sq_size						14h				
reserved						...						reserved						...		...		...										
reserved						sq_number_N						reserved						po	rt	VL		log2_sq_size										

It is SW responsibility to make sure that the number of QPs and DCRs on this schedule queue will not exceed its size.

Table 873 - SQ_INTERNAL_CFG - Schedule Queue Configuration Register Fields

Offset	Bits	Name	Description	R/W/ Index
		v	valid entry	RW
		log_sq_size	log2 of SQ size (in units of QPs that will can use this SQ.)	RW

28.6 MISC Registers

28.6.1 Loopback Control Register

Controls whether loopback packets for the same HCA are sent to the internal loopback interface, or are sent to wire. This command can only be written to prior to executing INIT_HCA command. Register may be read at any time.

Table 874 - Loopback Control Register

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
no lb	reserved							port																							00h	
																														04h		
																														08h		
																														0Ch		

Table 875 - Loopback Control Register Fields

Offset	Bits	Name	Description	R/W/ Index
00h	31	no_lb	When set, the HCA will always send packets meant for the same port to wire, rather than perform internal loopback	RW
	23:16	port	1 - port 1 2 - port 2 255 - both ports otherwise - reserved	
04h	31:0	reserved		
08h	31:0	reserved		
0Ch	31:0	reserved		

28.6.2 DB_THRESHOLD Register

Enables setting threshold on doorbell FIFO. After setting threshold $\neq 0$, when FIFO reaches this threshold an asynchronous event will be sent (See “Doorbell / Blueflame Congestion Event” on page 499). FW will then set threshold to maximum. DB FIFO size should then be polled by SW. When FIFO size reaches required size, the DB_THRESHOLD register should be used to restore upper threshold. This mechanism can be used both for blue flame buffer and slow path doorbell FIFO, as specified in the type field.

Table 876 - DB_THRESHOLD - Argument Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									
																														type
																														00h

Table 877 - DB_THRESHOLD - Argument Field Descriptions

Offset	Bits	Name	Description	Access
00h	0	type	0 - slow path door bell FIFO 1 - blue flame buffer	

Table 878 - DB_THRESHOLD Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																threshold												00h				
																size												04h				
																																08h
																																0Ch

Table 879 - DB_THRESHOLD Register Field Description

Offset	Bits	Name	Description	Access
00h	15:0	threshold	Required upper watermark. When FIFO/buffer reaches this size- an event will be sent	RW
04h	15:0	size	Current FIFO/buffer size	RO
08h	31:0	reserved		
0Ch	31:0	reserved		

28.7 Unit Performance Counters Registers

vport has several groups of performance counters to enable measuring and debug of chip performance. Each performance counter block is comprised of 8 32-bit counters which can count up to 8 different events. Alternatively, any consecutive pair of 32-bit counters (starting with even index) can be merged into a single 64 bit counter. Configuring performance counter event also resets the counter.

The unit performance counters should be activated as follows:

1. Unit performance counter block should be activated using the UNIT_PERF_COUNTER_GLOBAL_CONTROL register (See Section 28.7.1, “UNIT_PERF_COUNTER_GLOBAL_CTRL Register,” on page 1098).
2. Unit performance counter block should be configured using the UNIT_PERF_COUNTER_CFG register (See Section 28.7.2, “UNIT_PERF_COUNTER_CFG Register,” on page 1099).

3. If performance counters need to be read atomically (i.e a single snapshot of all 8 counters), unit performance counter block should now be deactivated using the UNIT_PERF_COUNTER_GLOBAL_CONTROL register (See Section 28.7.1, “UNIT_PERF_COUNTER_GLOBAL_CTRL Register,” on page 1098).
4. Unit performance counter data can be collected using the UNIT_PERF_COUNTERS_DATA register (See Section 28.7.3, “UNIT_PERF_COUNTERS_DATA Register,” on page 1101).

The unit performance counter used for all above registers are listed in Table 882, “Unit Performance Counters Enumeration” below. Performance counter is given as part of the ACCESS_REG “argument” field, See Table 1784, “ACCESS_REGISTER Input Structure Layout,” on page 1857.

Table 880 - Unit Performance Counters Argument Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								unit_perf_counter_number				0h				

Table 881 - Unit Performance Counters Argument Field Descriptions

Offset	Bits	Name	Description	Access
00h	7:0	unit_perf_counter_number	See Table 882, “Unit Performance Counters Enumeration,” on page 1097	

Table 882 - Unit Performance Counters Enumeration

Unit perf counter number	register name
0x01	RXB
0x02	RXT
0x03	RXW
0x04	RXS
0x05	RXC
0x06	SXD
0x07	SXW
0x08	SXP
0x09	SHU
0x0A	DPR
0x0B	DPS

Table 882 - Unit Performance Counters Enumeration

Unit perf counter number	register name
0x0C	TPT
0x0D	QPC
0x0E	ICMC_INPUT
0x0F-0x16	ICMC_BANKS (0-7)
0x17	ICMC_MISS_MACHINES
0x18	ICMC_OUTPUT
0x19	PXT
0x1A	PXDP
0x1B	IRISC

28.7.1 UNIT_PERF_COUNTER_GLOBAL_CTRL Register

This register enables stop and restart counting on all 8 counters in unit performance counter block simultaneously. This ensures retrieving coherent performance counters status.

Table 883 - UNIT_PERF_COUNTERS_GLOBAL_CTRL Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								cmd				00h				
																																04h-0Ch

Table 884 - UNIT_PERF_COUNTERS_GLOBAL_CTRL Register Fields Description

Offset	Bits	Name	Description	Access
00h	7:0	cmd	0x0 - stop counting in all performance counters 0x1 - start counting in all performance counters Otherwise - reserved	WO
04h-0Ch	96	reserved		

28.7.2 UNIT_PERF_COUNTER_CFG Register

This register sets events to be counted by the unit performance counter block specified in the `unit_perf_counter_number` field. Up to 8 events can be counted using 32-bit counters. For each individual 32-bit counter, a specific event can be specified (which also resets the counter), or configuration can be skipped. Additionally, two adjacent 32-bit counters (starting with even counter) can be merged into a single 64-bit counter.

Table 885 - UNIT_PERF_COUNTERS_CFG Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																cnt0_cmd								cnt0_select				00h				
																cnt1_cmd								cnt1_select				04h				
																cnt2_cmd								cnt2_select				08h				
																cnt3_cmd								cnt3_select				0Ch				
																cnt4_cmd								cnt4_select				10h				
																cnt5_cmd								cnt5_select				14h				
																cnt6_cmd								cnt6_select				18h				
																cnt7_cmd								cnt7_select				1Ch				

Table 886 - UNIT_PERF_COUNTERS_CFG Register Fields Description

Offset	Bits	Name	Description	Access
00h	15:12	cnt0_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt0_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW

Table 886 - UNIT_PERF_COUNTERS_CFG Register Fields Description

Offset	Bits	Name	Description	Access
04h	15:12	cnt1_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt1_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW
08h	15:12	cnt2_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt2_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW
0Ch	15:12	cnt3_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt3_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW
10h	15:12	cnt4_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt4_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW

Table 886 - UNIT_PERF_COUNTERS_CFG Register Fields Description

Offset	Bits	Name	Description	Access
14h	15:12	cnt5_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt5_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW
18h	15:12	cnt6_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt6_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW
1Ch	15:12	cnt7_cmd	On read: reserved. On write: 0=SKIP: do not modify this counter selector 1=SET: set this counter selector, and reset the counter	RW
	7:0	cnt7_select	Counter selector configuration. If set to 0 on even-numbered selectors, extends the consecutively (odd-numbered) counter to 64 bits. If set to 0 on odd numbered selectors, count core clock cycles on this counter. Otherwise selects a counter with specified ID.	RW

28.7.3 UNIT_PERF_COUNTERS_DATA Register

Allows reading counters for unit performance counter block specified in the unit_perf_counter_number field. If all 8 counters need to be read atomically - counters should first be stopped using the UNIT_PERF_COUNTER_GLOBAL_CTRL register (See Section 28.7.1, “UNIT_PERF_COUNTER_GLOBAL_CTRL Register,” on page 1098)

Table 887 - UNIT_PERF_COUNTERS_DATA Register layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
counter0																																00h

Table 887 - UNIT_PERF_COUNTERS_DATA Register layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
counter1																																04h
counter2																																08h
counter3																																0Ch
counter4																																10h
counter5																																14h
counter6																																18h
counter7																																1Ch

Table 888 - UNIT_PERF_COUNTERS_DATA Register Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	counter0	Performance counter0 data	RO
04h	31:0	counter1	Performance counter1 data	RO
08h	31:0	counter2	Performance counter2 data	RO
0Ch	31:0	counter3	Performance counter3 data	RO
10h	31:0	counter4	Performance counter4 data	RO
14h	31:0	counter5	Performance counter5 data	RO
18h	31:0	counter6	Performance counter6 data	RO
1Ch	31:0	counter7	Performance counter7 data	RO

28.7.4 MISC_COUNTERS Register

The MISC_COUNTERS register is used to read and write to several counters not exposed elsewhere. Only the value of 0x0 can be written to counters.

Table 889 - MISC_COUNTERS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
counter_select																																04h
																																08h

Table 889 - MISC_COUNTERS Register Layout

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0																	0Ch
ecc																																10h						
ldb_silent_drop																																14h-1Ch						

Table 890 - MISC_COUNTERS Register Fields

Offset	Bits	Name	Description	R/W/ Index
00h	31:0	reserved		
04h	31:0	counter_select	When performing a WRITE operation - , the counter select bitmask determines which registers are set: Bit 0 - ECC Bit 1 - LDB silent drop	
08h	31:0	reserved		
0Ch	31:0	ecc	Counts number of arrays which encountered a bit error	
10h	31:0	ldb_silent_drop	Counts number of packets silently dropped due to LDB error	
14h-1Ch	96	reserved		

28.8 FPGA Registers

This section describes the registers used to manage and configure FPGA device.

28.8.1 FPGA_CAP Register

FPGA_CAP register is used to report the capabilities of the FPGA device.

Table 891 - FPGA_CAP Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
fpga_id								fpga_device																								

Table 891 - FPGA_CAP Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															A0h-FCh

Table 892 - FPGA_CAP Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	fpga_id	FPGA identifier. 0x0: Newton_X 0x1: Edison	RO
	23:0	fpga_device	FPGA device type 0x0: UNKNOWN_DEVICE 0x1: KU040 0x2: KU060 0x3: KU060_2	RO
04h	31:0	register_file_ver	Shell registers space file version.	RO
08h	31	fpga_ctrl_modify	If set, SW is allowed to modify FPGA_CTRL register. See Table 895 , “FPGA_CTRL Register Layout,” on page 1108	RO
	30	flash_gw_lock	If set, Flash_GW_Lock and Flash_GW_UnLock operations in FPGA_CTRL are supported.	
	29	disconnect_fpga	If set, DISCONNECT and CONNECT operations in FPGA_CTRL are supported.	
	25:24	access_reg_query_mode	Indicates the QUERY operation’s permissions of FPGA_ACCESS_REG. [Internal] - In future can have a mode per range(VF). 0x0: Not_allowed - Query registers by FPGA_ACCESS_REG not allowed. 0x1: All_range_allowed- Query registers by FPGA_ACCESS_REG allowed for all addresses.	RO
	17:16	access_reg_modify_mode	Indicates the MODIFY operation’s permissions of FPGA_ACCESS_REG. [Internal] - In future can have a mode per range(VF). 0x0: Not_allowed - modify registers by FPGA_ACCESS_REG not allowed. 0x1: All_range_allowed- modify registers by FPGA_ACCESS_REG allowed for all addresses.	RO
10h	31:0	image_version	Image version.	RO

Table 892 - FPGA_CAP Register Fields

Offset	Bits	Name	Description	Access
14h	31:0	image_date	Creation date of current running image. Format: DDMMYYYY. For example: 0x12011995 means 12/01/1995 in DD/MM/YY.	RO
18h	31:0	image_time	Creation time of current running image. Format: 00HHMMSS. For example: 0x00015324 means 01:53:24 in HH:MM:SS	RO
1Ch	31:0	shell_version	Shell image version.	RO
30h-6Ch	512	shell_caps (See Table 893, “Shell_Caps Lay-out,” on page 1107)	Shell capabilities.	RO
70h	23:0	ieee_vendor_id	IEEE Vendor ID of sandbox user. 0x2C9: Mellanox	RO
74h	31:16	sandbox_product_version	Sandbox version.	RO
	15:0	sandbox_product_id	Sandbox product ID. For Mellanox sandbox products 0x0: unknown 0x1: example 0x2: IPsec 0x3: TLS	RO
78h	31:0	sandbox_basic_caps	Sandbox basic capabilities per sandbox product ID. For Mellanox sandbox products, see Table 796, “IPsec_Basic_Capabilities Structure Layout,” on page 1014.	RO
7Ch	15:0	sandbox_extended_caps_len	The length (in DWORDs) of sandbox extended capability (equals 0 when such capability is not present).	RO
80h-84h	64	sandbox_extended_caps_addr	Extended capabilities address. For Mellanox sandbox products, see Table 798, “IPsec_Extended_Capabilities Structure Layout,” on page 1014.	RO
88h-8Ch	64	fpga_ddr_start_addr	Start address of DDR	
90h-94h	64	fpga_cr_space_start_addr	Start address of cr-space	
98h	31:0	fpga_ddr_size	Size of DDR in a granularity of 1KB	
9Ch	31:0	fpga_cr_space_size	Size of cr-space in a granularity of 1KB	

Table 893 - Shell_Caps Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
max_num_qps																								total_rcv_credits								
																																04h
																																08h
																																0Ch
																																10h-3Ch

Table 894 - Shell_Caps Fields

Offset	Bits	Name	Description	Access
00h	31:16	max_num_qps	Maximum number of supported QPs.	RO
	7:0	total_rcv_credits	Total size of shared QP buffer size in granularity of 2 KB.	RO
04h	17:16	qp_type	Bit mask indicates which QP types are supported Bit 0: Shell_qp - Shell QP supported Bit 1: Sandbox_qp - Sandbox QP supported	RO
	10	rae	If set, RDMA atomic operation is supported.	RO
	9	rwe	If set, RDMA WRITE operation is supported.	RO
	8	rre	If set, RDMA READ operation is supported.	RO
	3	dc	If set, DC transport service is supported.	RO
	2	ud	If set, UD transport service is supported.	RO
	1	uc	If set, UC transport service is supported.	RO
	0	rc	If set, RC transport service is supported.	RO
08h	5:0	log_ddr_size	Log (base 2) of the DDR size in granularity of GB.	RO
0Ch	31:0	max_fp-ga_qp_msg_size	The maximum size of message size supported by FPGA Shell QP.	RO

28.8.2 FPGA_CTRL Register

FPGA_CTRL register is used to report the capabilities of the FPGA device.

The FPGA_CTRL register allows load and reset of FPGA images as explained in initialization flow. TBD (add cross reference).

This register is asynchronous. That is, after a load or reset operation, SW should perform QUERY and check the status field. It is recommend to not perform more than a single operation before checking the status of the previous one.

Table 895 - FPGA_CTRL Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
error_type								operation																status								00h
								flash_select_admin																flash_select_oper								04h
																																08h-0Ch

Table 896 - FPGA_CTRL Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	error_type	Error type. Valid only when status = failure(0x1) 0x0: Error_unknown 0x1: fpga_done_timeout 0x2: gpio_sig_mismatch 0x3: ddr_bist_timeout 0x4: sadb_reset_done_timeout 0x5: sadb_reset_success_timeout 0x6: ips_flush_done_timeout 0x7: flash_gw_lock_timeout 0x8: flash_gw_unlock_error 0x9: i2c_access_during_fpga_load 0xA: fpga_user_img_done_timeout	
	23:16	operation	Indicates the control operation to be performed. Allowed only when FPGA_CAP.fpga_ctrl_modify==1. See Table 891, “FPGA_CAP Register Layout,” on page 1103. 0x1: LOAD - when set, the FPGA will be forced to reload the image from flash according to image_select_admin value. 0x2: RESET - when set, the FPGA internal logic state (BRAMs, FFs, etc.) will be reset to the original state at loading time. 0x3: FLASH_SELECT - when set, map the flash GW according to flash_select_admin value. 0x4: Sandbox_Bypass_On - If set, sandbox logic is inactive until Sandbox_Bypass_Off is set. Packets will skip the sandbox logic. 0x5: Sandbox_Bypass_Off - If set, sandbox logic is active. Packets will be processed by the sandbox logic. 0x6: Reset_Sandbox - If set, the FPGA sandbox logic state (BRAMs, FFs, etc.) will be reset to the original state at loading time (no shell state is affected). 0x7: Flash_GW_Lock - lock the flash gateway. Users must take this lock before attempting to reads/writes to flash. Supported only when FPGA_CAP.flash_gw_lock==1. 0x8: Flash_GW_Unlock - Unlock flash gateway. Users must release this lock after completing their reads/writes to flash. Supported only when FPGA_CAP.flash_gw_lock==1. 0x9: DISCONNECT - Disable the ConnectX to FPGA management interface. This mode is useful when upgrading the FPGA via JTAG. supported only when FPGA_CAP.disconnect_fpga==1. Note that on this mode no other operation can be performed on the FPGA. 0xA: CONNECT - Enable the ConnectX to FPGA management interface.supported only when FPGA_CAP.disconnect_fpga==1.	WO
	7:0	status	Status of the last operation. 0x0: Success 0x1: Failure 0x2: In_progress 0x3: DISCONNECTED - FPGA is disconnect by DOSCONNECT operation. Valid only for query operation.	RO

Table 896 - FPGA_CTRL Register Fields

Offset	Bits	Name	Description	Access
04h	23:16	flash_select_admin	Used to select next flash to be used to load the images. 0x0: User 0x1: Factory_default Valid only for FLASH_SELECT and LOAD operations.	RW
	7:0	flash_select_oper	Current flash that is used to load images to FPGA. 0x0: User 0x1: Factory_default 0x2: Factory_failover - The device tried to load the user image, but due to a failure loaded the factory image. Valid only if status is ok.	RO

28.8.3 FPGA_ACCESS_REG_SPACE Register

This register is used to perform read and write operations from/to FPGA internal registers space. Before using this register, SW should read the allowed permission capabilities reported by `FPGA_CAP.access_reg_query_mode` and `FPGA_CAP.access_reg_modify_mode`. See Table 891, “FPGA_CAP Register Layout,” on page 1103.

Table 897 - FPGA_ACCESS_REG_SPACE Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
																size												04h				
address																												08h-0Ch				
data[0]								data[1]								data[2]								data[3]								10h
data[4]								data[5]								data[6]								data[7]								14h
...																												18h-..				

Table 898 - FPGA_ACCESS_REG_SPACE Register Fields

Offset	Bits	Name	Description	Access
04h	15:0	size	Number of bytes to read/write. Must be aligned to 4 bytes. Note that the maximum size supported is 64 bytes.	RW

Table 898 - FPGA_ACCESS_REG_SPACE Register Fields

Offset	Bits	Name	Description	Access
08h-0Ch	64	address	Allowed address depending on the FPGA_CAP.access_reg_query_mode and FPGA_CAP.access_reg_modify_mode. See Table 891, “FPGA_CAP Register Layout,” on page 1103.	RW
10h-..	31:24	data[...]	For Query operations, specify the read data from FPGA registers space. For Write operation, specify the data to be written to FPGA registers space.	RW

28.8.4 FPGA_SHELL_COUNTERS Register

FPGA_SHELL_COUNTERS register is used to query general FPGA shell counters.

Table 899 - FPGA_SHELL_COUNTERS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
clear																																04h
																																08h-0Ch
ddr_read_requests																																10h-14h
ddr_write_requests																																18h-1Ch
ddr_read_bytes																																20h-24h
ddr_write_bytes																																28h-2Ch
																																30h-6Ch

Table 900 - FPGA_SHELL_COUNTERS Register Fields

Offset	Bits	Name	Description	Access
04h	31	clear	Clear counters. Setting the clear bit will reset all counters.	OP

Table 900 - FPGA_SHELL_COUNTERS Register Fields

Offset	Bits	Name	Description	Access
10h-14h	64	ddr_read_requests	Total amount of DDR read requests.	RO
18h-1Ch	64	ddr_write_requests	Total amount of DDR write requests.	RO
20h-24h	64	ddr_read_bytes	Total amount of DDR read bytes.	RO
28h-2Ch	64	ddr_write_bytes	Total amount of DDR write bytes.	RO

28.9 Device Stress Registers

28.9.1 STRS_CAP Register

Table 901 - STRS_CAP Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
																																04h-3Ch

Table 902 - STRS_CAP Register Fields

Offset	Bits	Name	Description	Access
00h	31	stop_toggle	When set, Stop Toggle stressor is supported	RO
	30	fault_inject	When set, Fault Injection stressor is supported	RO
	29	mini_flow	When set, Mini Flow stressor is supported	RO
	28	resource_reduce	When set, Resource Reduction stressor is supported	RO

28.9.2 STRS_STOP_TOGGLE Register

Table 903 - STRS_STOP_TOGGLE Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
supported	active	past active																														00h
																									type						04h	
polarity																			log_duty_cycle							log_stressor					08h	
																															0Ch	
per_type_modifier																															10h-2Ch	
																															30h-3Ch	

Table 904 - STRS_STOP_TOGGLE Register Fields

Offset	Bits	Name	Description	Access
00h	31	supported	Stesser of this type is supported	RO
	30	active	Stesser of this type is currently active	RO
	29	past_active	Stesser of this type is was active at some point in history	RO

Table 904 - STRS_STOP_TOGGLE Register Fields

Offset	Bits	Name	Description	Access
04h	7:0	type	Type of mechanism to toggle 0x0: DC_CNAK - stop FW machine which sends DC CNAK packets on DC LIFO empty 0x1: RXT_CHECKS - stop FW machine which handles RX transport check traps 0x2: TIMEOUT - stop FW machine which handles QP transport timeout events 0x3: SX_ERROR - stop FW machine which handles SX WQE or data gather errors 0x4: RX_ERROR - stop FW machine which handles all RX errors apart from RXT checks 0x5: MX_ERROR - stop FW machine which handles other transport errors, such as data signature error 0x6: MAD_TRAP - stop FW machine which sends MAD trap packets 0x7: RXT_SLICE - stop HW which loads RXT unit from RX buffers 0x8: QOS_ARBITER - stop HW which servers schedule queue pops / flicks 0x9: RXB_HANG - stop HW which loads RX buffers – per port / VL configuration 0xA: FW_SCHED_Q - stop FW machine which empties the FW schedule queue 0xB: LOCK_RESOURCE - stop FW machines attempting to take semaphores 0xC: IRISC_HANG - stop all FW processing 0xD: SXW_SLICE - stop HW SXW slices from executing WQE requests 0xE: RXC_CQE - stop HW posting CQEs 0xF: RXC_EQE - stop HW posting EQEs 0x10: SXP_HANG - stop HW SX from transmitting packets 0x11: SX_EXT_DB - stop HW processing of external doorbells (schedule queue push) 0x12: SX_INT_DB - stop HW processing of internal doorbells (responder ack / read response requests) 0x13: QPC_SLICE - stop HW QP context read requests Other values are reserved 0x14: RXB_HOST_HANG - Stop HW which loads RX buffers – per host / VL configuration. When supported deprecates RXB_HANG <u>0x15: DEVICE_FLUSH - stops the flow that validates flushing of unused contexts from deice engines (e.g. packets, ICM contexts)</u> <u>0x16 : PAUSE_TX - generates pause frames toward the network – per port & SL/priority configuration.</u>	Index

Table 904 - STRS_STOP_TOGGLE Register Fields

Offset	Bits	Name	Description	Access
08h	31	polarity	Indicates how the Stressor affects the device during the Stressor active time. 0x0: FLOW_STOPPED - HW/FW flow is stressed 0x1: FLOW_ACTIVE - HW/FW is flow is working normally The Stressor will have the opposite behavior when it is not active.	RW
	12:8	log_duty_cycle	The Stressor will be active for 50usec * 2 ^{log_stressor} period, once every 50usec * 2 ^{log_duty_cycle} .	RW
	4:0	log_stressor		RW
10h-2Ch	256	per_type_modifier	stressor Modifier according to <i>type</i> field. For RXB_HANG See Table 905, “RXB_HANG Stop Toggle Modifier Layout,” on page 1116 For LOCK_RESOURCE See Table 911, “LOCK_RESOURCE Stop Toggle Modifier Layout,” on page 1118 For SXP_HANG See Table 907, “SXP_HANG Stop Toggle Modifier Layout,” on page 1117 For RXB_HOST_HANG See Table 909, “RXB_HOST_HANG Stop Toggle Modifier Layout,” on page 1118 For PAUSE_TX See Table 919, “PAUSE_TX Stop Toggle Modifier Layout,” on page 1121	RW

Table 905 - RXB_HANG Stop Toggle Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
port_number																vl_tc_mask																

Table 906 - RXB_HANG Stop Toggle Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	port_number	Physical Port Number 1: port_number1 2: port_number2 Other values are reserved	WO
	15:0	vl_tc_mask	bitmap per Virtual Lane/Traffic Class bit 0: VL_TC_0 bit 1: VL_TC_1 bit 2: VL_TC_2 bit 3: VL_TC_3 bit 4: VL_TC_4 bit 5: VL_TC_5 bit 6: VL_TC_6 bit 7: VL_TC_7 bit 15: VL_TC_15 Other bits are reserved	RW

Table 907 - SXP_HANG Stop Toggle Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
																																04h-1Ch

Table 908 - SXP_HANG Stop Toggle Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	port_number	Physical Port Number 1: port_number1 2: port_number2 Other values are reserved	WO

Table 909 - RXB_HOST_HANG Stop Toggle Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
port_number														pnat	vl_tc_mask																	04h-1Ch

Table 910 - RXB_HOST_HANG Stop Toggle Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	port_number	Physical Port Number 1: port_number1 2: port_number2 Other values are reserved	WO
	17:16	pnat	Port number access type. must be set to 0x2	RW
	15:0	vl_tc_mask	bitmap per Virtual Lane/Traffic Class bit 0: VL_TC_0 bit 1: VL_TC_1 bit 2: VL_TC_2 bit 3: VL_TC_3 bit 4: VL_TC_4 bit 5: VL_TC_5 bit 6: VL_TC_6 bit 7: VL_TC_7 bit 15: VL_TC_15 Other bits are reserved	RW

Table 911 - LOCK_RESOURCE Stop Toggle Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
category																																04h-1Ch
category_modifier																																

Table 912 - LOCK_RESOURCE Stop Toggle Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	category	Resource type category 0x0: GENERAL_SEMAPHORE 0x1: ICM_RESOURCE 0x2: UAPP_RESOURCE Other values are reserved	RW
04h-1Ch	224	category_modifier	LOCK_RESOURCE Modifier according to <i>category</i> field For GENERAL_SEMAPHORE See Table 913, “GENERAL_SEMAPHORE Category Layout,” on page 1119 For ICM_RESOURCE See Table 915, “ICM_RESOURCE Category Layout,” on page 1120 For UAPP_RESOURCE See Table 917, “UAPP_RESOURCE Category Layout,” on page 1120	RW

Table 913 - GENERAL_SEMAPHORE Category Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-18h

Table 914 - GENERAL_SEMAPHORE Category Fields

Offset	Bits	Name	Description	Access
00h	7:0	type	Resource type 0x0: QPC_GW 0x1: CQE_GW 0x2: EQE_GW 0x3: MEM_GW 0x4: IPC Other values are reserved	RW
04h	31:24	log_toggle_cycle	While the stressor locks the resource, it will release it shortly every $2^{\log\_toggle\_cycle}$, given in usec. Maximal value is 1 sec (0x14)	RW

Table 915 - ICM_RESOURCE Category Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
1	0																															00h			
																									type										04h
log_toggle_cycle																gvmi												08h							
									index																		0Ch-18h								

Table 916 - ICM_RESOURCE Category Fields

Offset	Bits	Name	Description	Access
00h	7:0	type	TBD	RW
04h	31:24	log_toggle_cycle	While the stressor locks the resource, it will release it shortly every $2^{\text{log\_toggle\_cycle}}$, given in usec. Maximal value is 1 sec (0x14)	RW
	15:0	gvmi	ICM resource GVMI	RW
08h	23:0	index	ICM resource index	RW

Table 917 - UAPP_RESOURCE Category Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
1	0																																		
																									type										00h
log_toggle_cycle																gvmi															04h				
									index																			08h							
																												0Ch-18h							

Table 918 - UAPP_RESOURCE Category Fields

Offset	Bits	Name	Description	Access
00h	7:0	type	0x0: UAPP_QP 0x1: SRQ 0x2: UAPP_SRQ_META 0x3: UAPP_RES_CQ 0x4: UAPP_REQ_CQ 0x5: UAPP_EQ 0x6: NSQ 0x7: NCQ Other values are reserved	RW
04h	31:24	log_toggle_cycle	While the stressor locks the resource, it will release it shortly every $2^{\text{log\_toggle\_cycle}}$, given in usec. Maximal value is 1 sec (0x14)	RW
	15:0	gvmi	uApp resource GVMI	RW
08h	23:0	index	uApp resource index: for uapp QP enter QPN. for uapp SRQ, uapp SRQ meta, UAPP CQ enter the srq index for nvme SQ, nvme CQ enter the backend controller id	RW

Table 919 - PAUSE_TX Stop Toggle Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
port_number														pnat	sl_prio_mask																04h-1Ch	

Table 920 - PAUSE_TX Stop Toggle Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	port_number	Physical Port Number 1: port_number1 2: port_number2 Other values are reserved	WO
	17:16	pnat	Port number access type. must be set to 0	RW
	15:0	sl_prio_mask	bitmap per Virtual Lane/Traffic Class bit 0: SL_PRIO_0 bit 1: SL_PRIO_1 bit 2: SL_PRIO_2 bit 3: SL_PRIO_3 bit 4: SL_PRIO_4 bit 5: SL_PRIO_5 bit 6: SL_PRIO_6 bit 7: SL_PRIO_7 Other bits are reserved	RW

28.9.3 STRS_FAULT_INJECT Register

Table 921 - STRS_FAULT_INJECT Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
supported	active	past_active																														04h
																																08h
																																0Ch
																																10h-2Ch
																																30h-3Ch

Table 922 - STRS_FAULT_INJECT Register Fields

Offset	Bits	Name	Description	Access
00h	31	supported	Stesser of this type is supported	RO
	30	active	Stesser of this type is currently active	RO
	29	past_active	Stesser of this type is was active at some point in history	RO
04h	7:0	type	Type of fault to activate 0x0: ICM_ALLOC_REFUSE - cause FW to return FAIL on ICM allocation requests 0x1: ICM_ALLOC_BUSY - cause FW to return BUSY on ICM allocation request 0x2: EQE_GW_BUSY - cause FW to return busy when trying to access EQE GW 0x3: CQE_GW_BUSY - cause FW to return busy when trying to access CQE GW 0x4: RX_FENCE_BUSY - cause FW to return busy when checking RX fences done 0x5: SX_FENCE_BUSY - cause FW to return busy when checking SX fences done 0x6: RXT_SLICE_FENCE_BUSY - cause FW to return busy when checking RXT fences done 0x7: SXD_SLICE_FENCE_BUSY - cause FW to return busy when checking SXD fences done 0x8: GENERAL_FENCE_BUSY - cause FW to return busy when checking other fences done 0x9: SMBUS_FAILED - cause FW to indicate failure upon initiating SMBus transaction Other values are reserved	Index
08h	15:0	num_skip	The fault will be injected <i>num_repeat</i> consecutive times the relevant flow is called, and will not be injected for the following <i>num_skip</i> calls to the relevant flow. Value 0x0 for <i>num_repeat</i> indicates the Fault will not be injected.	RW
0Ch	15:0	num_repeat		RW
10h-2Ch	256	per_type_modifier (See Table 923, “SMBUS - FAILED Fault Inject Modifier Layout,” on page 1124)	stressor Modifier according to <i>type</i> field. For SMBUS_FAILED, See Table 923, “SMBUS_FAILED Fault Inject Modifier Layout,” on page 1124	RW

Table 923 - SMBUS_FAILED Fault Inject Modifier Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
																									smbus_gw										00h
																																04h-1Ch			

Table 924 - SMBUS_FAILED Fault Inject Modifier Fields

Offset	Bits	Name	Description	Access
00h	7:0	smbus_gw	SMBUS GW identifier	RW

28.9.4 STRS_MINI_FLOW Register

Table 925 - STRS_MINI_FLOW Register Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
1	0																															00h	
supported	active	past_active																															00h
																									type							04h	
																freq												08h					
																num_repeat												0Ch					
per_type_modifier																														10h-2Ch			
																														30h-3Ch			

Table 926 - STRS_MINI_FLOW Register Fields

Offset	Bits	Name	Description	Access
00h	31	supported	Stesser of this type is supported	RO
	30	active	Stesser of this type is currently active	RO
	29	past_active	Stesser of this type is was active at some point in history	RO
04h	7:0	type	Type of Mini Flow to activate 0x0: SX_FLICK_THROTTLE - enable / disable the SX flick mechanism 0x1: INVALIDATE_STEERING_CACHE 0x2: INVALIDATE_RXT_QP_L0_CACHE 0x3: INVALIDATE_DCT_L0_CACHE 0x4: INVALIDATE_LDB_REQSL_CACHE 0x5: INVALIDATE_RXC_CACHE 0x6: INVALIDATE_SXDC_CACHE 0x7: RECONSTRUCT_STEERING_BYPASS 0x8: INVALIDATE_LDB_CACHE 0x9: PCI_READ_ERROR - simulate a PCI error on Px polling machine read 0xA: INVALIDATE_ALL_RO_CACHES 0xB: INVALIDATE_PKEY_CACHE 0xC: INVALIDATE_GUID_CACHE 0xD: INVALIDATE_PORT_INFO_CACHE 0xE: INVALIDATE_QP_CACHE 0xF: IRISC_HANG 0x10: PACKET_DROP Other values are reserved	Index
08h	15:0	freq	The Mini flow will be injected num_repeat times, every 50 usec * freq .	RW
0Ch	15:0	num_repeat	Value 0x0 for freq indicates the Mini Flow will not be injected.	RW
10h-2Ch	256	per_type_modifier	stressor Modifier according to type field. For IRISC_HANG See Table 927, “IRISC_HANG Mini-Flow Modifier Layout,” on page 1125 For PACKET_DROP See Table 929, “PACKET_DROP Mini-Flow Modifier Layout,” on page 1126	RW

Table 927 - IRISC_HANG Mini-Flow Modifier Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
irisc_hang_mask																																00h
irisc_hang_cap_mask																																04h

Table 927 - IRISC_HANG Mini-Flow Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															08h-1Ch

Table 928 - IRISC_HANG Mini-Flow Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:0	irisc_hang_mask	Bit per Irisc. When set, IRISC will hang (The miniflow does not release the Irisc when done)	RW
04h	31:0	irisc_hang - cap_mask	Bit per Irisc. When set, the device supports causing the respective Irisc to hang using this miniflow.	RO

Table 929 - PACKET_DROP Mini-Flow Modifier Layout

3	3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1	0																															00h
port_number																																04h-1Ch

Table 930 - PACKET_DROP Mini-Flow Modifier Fields

Offset	Bits	Name	Description	Access
00h	31:24	port_number	Physical Port Number 1: port_number1 2: port_number2 Other values are reserved	RW
	3:0	num_packets	The number of dropped packets when the Miniflow is activated	RW

28.9.5 STRS_RESOURCE Register

Table 931 - STRS_RESOURCE Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
supported	active	past active																														00h
																									type						04h	
reduce_percentage																															08h	
																															0Ch-3Ch	

Table 932 - STRS_RESOURCE Register Fields

Offset	Bits	Name	Description	Access
00h	31	supported	Stesser of this type is supported	RO
	30	active	Stesser of this type is currently active	RO
	29	past_active	Stesser of this type is was active at some point in history	RO
04h	7:0	type	Type of resource to limit 0x0: SX_SLICE 0x1: RX_SLICE Other values are reserved	Index
08h	31:0	reduce_percent- age	Limit as a percentage of the available resources Value of 100 (0x64) indicates return to full utilization	RW

28.10 SECURE_HOST

Table 933 - SECURE_HOST Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
operation																																00h
																																04h
key																																08h-0Ch

Table 933 - SECURE_HOST Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																10h-3Ch

Table 934 - SECURE_HOST Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	operation	0x0: UNLOCK 0x1: LOCK	WO
08h-0Ch	64	key		WO

28.11 CORE DUMP Register

Please refer to Core Dump section (See [Section 27.9, “Core Dump,”](#) on page 1078).

Table 935 - CORE_DUMP Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
more_dump																																00h
																																core_dump_type
																																04h
																																08h
																																0Ch
																																10h
																																14h
																																18h-1Ch
																																20h-24h

Table 935 - CORE_DUMP Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																28h-3Ch

Table 936 - CORE_DUMP Register Fields

Offset	Bits	Name	Description	Access
00h	31	more_dump	If set, the device has additional information that has not been dumped yet. Valid when core_dump_type==CR_DUMP_TO_MEM.	RO
	7:0	core_dump_type	Core Dump type. 0x0: GENERAL - supported only when HCA_ - CAP.core_dump_general==1. 0x1: QP - supported only when HCA_ - CAP.core_dump_qp==1. 0x2: CR_DUMP_TO_MEM - Dumped to user allocated buffer. Supported only when HCA_ - CAP.log_cr_dump_to_mem_size!=0	WO
08h	15:0	vhca_id	vhca_id where the dumped QP allocated. valid only when core_dump_type==QP	WO
0Ch	23:0	qpn	QP number needed to be dumped. valid only when core_dump_type==QP	WO
10h	31:0	mkey	Memory key to dump to. Valid when core_dump_type==CR_DUMP_TO_MEM.	WO
14h	31:0	size	In write, the size of maximum allocated buffer that the device can use. In read, the actual written size. Valid when core_dump_type==CR_DUMP_TO_MEM.	RW
18h-1Ch	64	address	VA address (absolute address) of memory where to start dumping. Valid when core_dump_type==CR_DUMP_TO_MEM.	WO
20h-24h	64	cookie	a cookie provided by the device. SW shall read the cookie upon command done and shall provide it on the next call in case dump_more==1. On first call, cookie shall be 0. Valid when core_dump_type==CR_DUMP_TO_MEM.	RW

28.12 Resource DUMP Register

Please refer to Resource Dump section (See [Section 27.8, “Resource Dump,” on page 1062](#)).
This register is supported when HCA_CAP.resource_dump==1.

Table 937 - RESOURCE_DUMP Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
more_dump		inline_dump	vhca_id_valid									seq_num				segment_type																00h
																vhca_id																04h
index1																																08h
index2																																0Ch
num_of_obj1																num_of_obj2																10h
																																14h
device_opaque																																18h-1Ch
mkey																																20h
size																																24h
address																																28h-2Ch
inline_data[0]																																30h
...																																...
inline_data[51]																																FCCh

Table 938 - RESOURCE_DUMP Register Fields

Offset	Bits	Name	Description	Access
00h	31	more_dump	If set, the device has additional information that has not been dumped yet.	RW
	30	inline_dump	If set, data is dumped in the register in inline_data field. otherwise dump to mkey.	WO
	29	vhca_id_valid	If set, then vhca_id field is valid. Otherwise dump resources on my vhca_id.	WO
	19:16	seq_num	Sequence number. 0 on first call of dump and incremented on each more dump.	RW
	15:0	segment_type	See Section 27.8, “Resource Dump,” on page 1062.	RW
04h	15:0	vhca_id	vhca_id where the resource is allocated.	WO
08h	31:0	index1	First object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	RW
0Ch	31:0	index2	Second object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	RW
10h	31:16	num_of_obj1	The amount of objects to dump starting for index 1 SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj1_-supports_all is set, the special value of 0xffff represents “all”. When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents “active”. The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj1 is “0”.	RW
	15:0	num_of_obj2	The amount of objects to dump starting for index 2. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj2_-supports_all is set, the special value of 0xffff represents “all”. When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents “active”. The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj2 is “0”.	RW

Table 938 - RESOURCE_DUMP Register Fields

Offset	Bits	Name	Description	Access
18h-1Ch	64	device_opaque	An opaque provided by the device. SW shall read the device_opaque upon command done and shall provide it on the next call in case dump_more==1. On first call, device_opaque shall be 0.	RW
20h	31:0	mkey	Memory key to dump to. Valid when inline_dump==0.	WO
24h	31:0	size	In write, the size of maximum allocated buffer that the device can use. In read, the actual written size. In granularity of Bytes.	RW
28h-2Ch	64	address	VA address (absolute address) of memory where to start dumping. Valid when inline_dump==0.	WO
30h-FCh	31:0	inline_data[52]	Data that is dumped in case of inline mode. Valid when inline_dump==1.	RW

28.13 CAPI Registers

28.13.1 CAPI_CTRL Register

This register is used to enable/disable and configure the CAPI. This register supported only when ICMD_QUERY_CAPABILITY.capi==1. For more information, please refer to Section 22, “CAPI- Coherent Accelerator Programming Interface,” on page 1035.

Table 939 - CAPI_CTRL Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
version																																00h
																																04h
																																08h-0Ch

Table 939 - CAPI_CTRL Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dsn_ctrl																																10h-14h
dra																																18h-1Ch
capi_base																																20h-24h
log_capi_bar_size																																28h
																																2Ch

Table 940 - CAPI_CTRL Register Fields

Offset	Bits	Name	Description	Access
00h	31:0	version	HOST CAPI version, must be 1	RW
04h	2	owner		
	1	enable_translations	Enables CAPI translation requests. Needs to be set by the CAPI owner after CAPI is enabled on the HOST. Non-owner function will fail to set this bit if it is not set on the CAPI owner. If the CAPI owner clears this bit an internal FLR will be performed on all the other functions that have this bit set.	RW
	0	enable_invalidations	Enables Responses to CAPI invalidation request. Must be set before enabling CAPI on the host. The command that sets this bit must be issued by the CAPI owner and it must also set all the CAPI parameters in this registers. Non owner function will fail to set this bit if it is not set on the CAPI owner.	RW
10h-14h	64	dsn_ctrl	PSL data send control register. See XSL spec for more details	
18h-1Ch	64	dra	XSL Dummy read address. See XSL spec for more details	
20h-24h	64	capi_base	CAPI Protocol communication area address. This corresponds to the 64bit BAR defined by BARs 4/5 in the XSL spec. Must be aligned to the bar size	

Table 940 - CAPI_CTRL Register Fields

Offset	Bits	Name	Description	Access
28h	31:0	log_capi_bar_size	Log (base 2) of size of the CAPI BAR. must be >=12	

28.14 NCFG - NIC Configuration Register

This register is used to enable/disable device features.

This register is supported when ICMD_QUERY_CAPABILITY.ncfg_reg==1.

Table 941 - NCFG Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
field_select																																00h-04h	
																																roce_offload_disable	08h
																																	0Ch-3Ch

Table 942 - NCFG Register Fields

Offset	Bits	Name	Description	Access
00h-04h	64	field_select	In Query, indicates which fields are supported. In Write, indicates which field to modify. Bit 0x0: roce_offload_disable	RW
08h	0	roce_offload_disable	If set, then ROCE is disabled for PF and VFs/SFs belong to it.	RW

28.15 CRYPTO_OPERATIONAL Register

Crypto engines that are indicated to work in Wrapped mode in the capabilities field wrapped_import_method are in commissioning mode when out-of-the-factory and are unusable until the following register is used to switch them to Operational mode.

This register supplies the first import_KEK and the first Officer Credential in plaintext.

Software driver should make sure that wrapped crypto engines are operational by checking the wrapped_crypto_operational bit in Table 1625, “Crypto Capabilities Layout,” on page 1802. If drivers were running during the change to operational mode, drivers need to be restarted. The supplied import_KEK and Officer Credential are both stored at index 0, and can be used for further keys/credentials management.

For more information, please See Section 20.6, “Disk Encryption,” on page 1003.

Table 943 - CRYPTO_OPERATIONAL Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
wrapped_crypto_operational																																kek_size	00h
																																04h	
credential																																08h-2Ch	
kek																																30h-4Ch	
																																50h-7Ch	

Table 944 - CRYPTO_OPERATIONAL Register Fields

Offset	Bits	Name	Description	Access
00h	31	wrapped_crypto_operational	If set to 1, device is in operational mode. Read only.	RO
	3:0	kek_size	0x0: SIZE_128 0x1: SIZE_256	WO

Table 944 - CRYPTO_OPERATIONAL Register Fields

Offset	Bits	Name	Description	Access
08h-2Ch	320	credential	Plaintext credential secret for non-volatile index 0	WO
30h-4Ch	256	kek	Plaintext import_KEK for non-volatile index 0	WO

28.16 CRYPTO_COMMISSIONING Register

When executed while driver and users are running, all existing Login Objects become invalid and all keys are destroyed. Hence, all in-progress or future crypto operations fail.

For more information, please see Section 20.6, “Disk Encryption,” on page 1003.

Table 945 - CRYPTO_COMMISSIONING Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
token																																00h-TBD

Table 946 - CRYPTO_COMMISSIONING Register Fields

Offset	Bits	Name	Description	Access
00h-TBD	TBD	token	A secret token from Mellanox to allow re-commissioning of the device. Note that all previously loaded keys will be removed.	WO

28.17 IMPORT_KEK_HANDLE Register

28.17.1 Usage

- For query operation, use Read.

When used for query, the kek_object will not include the KEK itself, only state and size. In case the provided KEK ID does not exist, a failure is returned.

- For add/delete operations, use Write. The Write_operation field would then determine whether it is an "add" or "delete" request.

When used for delete, the import_KEK being deleted cannot be used as the import_KEK wrapping the login credential. If a running process/driver has an active Login object that was using

the deleted import_KEK, that Login object becomes invalid and sends an async event to announce it. The user would need to destroy and re-create a new Login object with another import_KEK.

For more information, please see Section 20.6, “Disk Encryption,” on page 1003.

Table 947 - IMPORT_KEK_HANDLE Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
crypto_login_object(Table 302, “CRYPTO_LOGIN Object Format,” on page 570)																																00h-FCh
import_kek_object(Table 297, “IMPORT_KEK Object Format,” on page 567)																																100h-1FCh
				write_operation				import_kek_id																								200h
																																204h-2FCh

Table 948 - IMPORT_KEK_HANDLE Register Fields

Offset	Bits	Name	Description	Access
00h-FCh	2048	crypto_login_object (Table 302, “CRYPTO_LOGIN Object Format,” on page 570)	Information required to identify an officer. modify_field_select is reserved. Needed for all operations.	
100h-1FCh	2048	import_kek_object (Table 297, “IMPORT_KEK Object Format,” on page 567)	The import_KEK to add. For add, this is input (WO) of new KEK info. For query, this output (RO) with limited info of KEK. modify_field_select is reserved. For delete this field is reserved.	
200h	27:24	write_operation	0x1: ADD_OPERATION 0x2: DELETE_OPERATION	WO
	23:0	import_kek_id	For query/delete ops, this is input (WO). For add, input to this field is ignored, output is the selected ID.	INDEX

28.18 CREDENTIAL_HANDLE Register

28.18.1 Usage

- For query operation, use Read.

When used for query, the credential_object will not include the credential itself, only state and role. In case the provided credential ID does not exist, a failure is returned.

- For add/delete operation, use Write. The write_operation field would then determine whether it is an "add" or "delete" request.

When used for delete, the credential being deleted can't be used in the login_object. If a running process/driver has an active Login object that was using the deleted credential, that Login object becomes invalid and sends an async event to announce it. The user would need to destroy and re-create a new Login object with different credentials.

For more information, please See Section 20.6, "Disk Encryption," on page 1003.

Table 949 - CREDENTIAL_HANDLE Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
crypto_login_object(Table 302, “CRYPTO_LOGIN Object Format,” on page 570)																																00h-FCh
credential_object(Table 300, “CREDENTIAL Object Format,” on page 569)																																100h-1FCh
				write_operation				credential_id																								200h
																																204h-2FCh

Table 950 - CREDENTIAL_HANDLE Register Fields

Offset	Bits	Name	Description	Access
00h-FCh	2048	crypto_login_object (Table 302, "CRYPTO_LOGIN Object Format," on page 570)	Information required to identify an officer. modify_field_select is reserved. Needed for all operations.	
100h-1FCh	2048	credential_object(Table 300, "CREDENTIAL Object Format," on page 569)	The import_KEK to add. For add, this is input (WO) of new KEK info. For query, this output (RO) with limited info of KEK. modify_field_select is reserved. For delete this field is reserved.	

Table 950 - CREDENTIAL_HANDLE Register Fields

Offset	Bits	Name	Description	Access
200h	27:24	write_operation	0x1: ADD_OPERATION 0x2: DELETE_OPERATION	WO
	23:0	credential_id	For query/delete ops, this is input (WO). For add, input to this field is ignored, output is the selected ID.	INDEX

28.19 Network Ports Registers

This section lists the various registers that uses to query and modify the physical ports parameters and capabilities. The table below summarizes the port control registers.

[Internal] Use only registers 0x5000-0x5fff. Don't use 0x5020 and 0x5021.

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event		InfiniBand	Ethernet	
PREG	TBD	v			v	v	v	TBD
PCAM	0x507F	v			v	v	v	See Table 952, "PCAM - Ports Capabilities Mask Register Layout," on page 1146
PCAP	0x5001	v	v		v	v		See Table 954, "PCAP - Ports Capabilities Register Layout," on page 1148
PMLP	0x5002	v	v		v Only Query supported	v	v	See Table 956, "PMLP - Ports Module to Local Port Register Layout," on page 1149
PMTU	0x5003	v	v		v	v	v	See Table 960, "PMTU - Port MTU Register Layout," on page 1151
PTYS	0x5004	v	v		v	v	v	See Table 962, "PTYS - Port Type and Speed Register Fields," on page 1152
PPAD	0x5005	v	v		v		v	See Table 964, "PPAD - Port Physical Address Register Layout," on page 1157
PAOS	0x5006	v	v		v	v	v	See Table 966, "PAOS - Ports Administrative and Operational Status Register Layout," on page 1158
PPAOS	0x5040	v	v		v	v	v	See Table 968, "PPAOS - Port Phy Administrative and Operational Status Register Layout," on page 1160
PMAOS	0x5012	v	v		v	v	v	See Table 970, "PMAOS - Ports Administrative and Operational Status Register Layout," on page 1161
PFCC	0x5007	v	v		v		v	See Table 974, "PFCC - Ports Flow Control Configuration Register Layout," on page 1165
PPCNT	0x5008	v	v		v	v	v	See Table 976, "PPCNT - Ports Performance Counters Register Layout," on page 1171

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event		InfiniBand	Ethernet	
PUDE	0x5009	v		v		v	v	See Table 1018, “PUDE - Port Up/Down Event Layout,” on page 1224
PMPE	0x5024	v		v		v	v	See Table 1020, “PMPE - Port Module Plug/Unplug Event Layout,” on page 1226
PMPC	0x501F	v		v		v	v	See Table 1022, “PMPC – Port Module Present Change Register Layout,” on page 1227
PLIB	0x500A	v	v		TBD	v	v	See Table 1024, “PLIB - Port Local-Port to IB Port Mapping Register Layout,” on page 1228
PFSC	0x5043	v	v			v	v	See Table 1026, “PFSC - Port Forward Switching Control Register Layout,” on page 1229
PMMP	0x5044	v	v		v	v	v	See Table 1028, “PMMP - Port Module Memory Map Properties Register Layout,” on page 1230
PMCR	0x5045	v	v		v	v	v	See Table 1034, “PMCR - Port Module Control Register Layout,” on page 1237
PGMR	0x5069	v	-		v	v	v	See Table 1036, “PGMR- Port and Group Mapping Register Layout,” on page 1241
PMDR	0x503C	v	-		v	v	v	See Table 1038, “PMDR- Port Mapping Definitions Register Layout,” on page 1242
PMMIC	0x503D	v	v		v	v	v	See Table 1040, “PMMIC - Port Module Management Initialization Checker Register Layout,” on page 1244
Buffer Management Registers								
PPTB	0x500B	v	v		v		v	See Table 1044, “PPTB - Port Prio To Buffer Register Layout,” on page 1247
PBMC	0x500C	v	v		v		v	See Table 1046, “PBMC - Port Buffer Management Control Register Layout,” on page 1249
PBSR	0x5038	v	v		v[internal]		v	See Table 1050, “PBSR - Port Buffer Status Register Layout,” on page 1252
PSPA	0x500D	v	v			v	v	See Table 1054, “PSPA - Port Switch Partition Allocation Register Layout,” on page 1254
PELC	0x500E	v	v				v	See Table 1056, “PELC - Port Extended Link Capabilities Register Layout,” on page 1255

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event		InfiniBand	Ethernet	
PMLPN	0x5010	v	v			TB D	TB D	See Table 1058, “PMLPN - Port Mellanox Link Protocol Negotiation Register Layout,” on page 1259
PVLC	0x500F	v	v		v Only Query supported	v		See Table 1060, “PVLC - Port Virtual Lane Capabilities Register Layout,” on page 1260
PPSC	0x5011	v	v		v[i nter nal]	v	v	See Table 1062, “PPSC - Port Power Save Control Register Layout,” on page 1262
PLBF	0x5015	v	v			v	v	See Table 1064, “PLBF - Port Loopback Filter Register Layout,” on page 1264
PIFR	0x5016	v	v			v	v	See Table 1066, “PIFR - Port Isolation Filter Layout,” on page 1265
PIPG	0x5017	v	v		v[i nter nal]		v	See Table 1068, “PIPG - Port Inter Packet Gap Register Layout,” on page 1266
PPLR	0x5018	v	v		v	v	v	See Table 1070, “PPLR - Port Physical Loopback Register Layout,” on page 1267
PMPR	0x5013	v	v		v[i nter nal]	v	v	See Table 1072, “PMPR - Port Module Properties Register Layout,” on page 1268
PEIR	0x5019	v	v		v[i nter nal]	v	v	See Table 1074, “PEIR - Port Error Injection Register Layout,” on page 1269
PLPC	0x5022	v	v		v	v	v	See Table 1082, “PLPC - Phy Link Profiles Control Register Layout,” on page 1274
PPLM	0x5023	v	v		v	v	v	See Table 1084, “PPLM - Port Phy Link Mode Register Layout,” on page 1277
PPAMP	0x5025	v	v		v	v	v	See Table 1088, “[Internal] PPAMP - Port Phy opAMP Data Register Layout,” on page 1286

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event		InfiniBand	Ethernet	
SLRP	0x5026	v	v		v	v	v	See Table 1090, “SLRP - Serdes Lane Receive Parameters Register Layout,” on page 1288
SLRIP	0x5057	v	v		v	v	v	See Table 1100, “[Internal] Serdes Lane Receive Input Buffer Parameters Register Layout,” on page 1298
SLRD	0x5042	v	v		v	v	v	See Table 1102, “SLRD - Serdes Lane Receive Defect Register Layout,” on page 1300
SLTP	0x5027	v	v		v	v	v	See Table 1104, “SLTP - Serdes Lane Transmit Parameters Register Layout,” on page 1302
SLUTP	0x506A	v	v		v	v	v	See Table 1112, “SLUTP - Serdes Lane USR Transmit Parameters Register Layout,” on page 1308
SLRG	0x5028	v	v		v	v	v	See Table 1114, “SLRG - Serdes Lane Receive Grade Register Layout,” on page 1310
SLREG	0x5058	v	v		v	v	v	See Table 1120, “SLREG - Serdes Lane Receive Eye Grade Register Layout,” on page 1315
SLCCT	0x503B	v	v		v	v	v	See Table 1130, “SLCCT - Serdes Lane CTLE Calibration Table Register Layout,” on page 1325
PTAS	0x5029	v	v		v	v	v	See Table 1134, “PTAS - Phy Tuning Algorithm Settings Register Layout,” on page 1327
PPLL	0x5030	v	v		v	v	v	See Table 1136, “PPLL - Phy PLL Status Register Layout,” on page 1330
PHBR	0x5034	v	v		v[internal]	v	v	See Table 1146, “PHBR - Port Histogram Binding Register Layout,” on page 1334
PHCR	0x503A	v	v		v[internal]	v	v	See Table 1151, “PHCR - Port Histogram Configuration Register Layout,” on page 1337
PHRR	0x5035	v	v		v[internal]	v	v	See Table 1155, “PHRR - Port Histogram Reading Register Layout,” on page 1339
PPTT	0x5036	v	v		v	v	v	See Table 1157, “PPTT - Port PRBS Transmit Test Register Layout,” on page 1340
PPRT	0x5037	v	v		v	v	v	See Table 1159, “PPRT - Port PRBS Receive Test Register Layout,” on page 1346

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event		InfiniBand	Ethernet	
PCMR	0x5041	v	v		v	v	v	See Table 1167, “PCMR - Ports Check Mask Register Layout,” on page 1357
PDDR	0x5031	v			v	v	v	See Table 1169, “PDDR - Port Diagnostics Database Register Layout,” on page 1362
PCNR	0x5050		v			v	v	See Table 1086, “PCNR- Phy CoNfiguration Register Layout,” on page 1285
PPBMP	0x5051	v	v			v	v	See Table 1209, “PPBMP - Port Phy BER Monitor Parameters Register Layout,” on page 1404
PPBMC	0x5052	v	v			v	v	See Table 1213, “PPBMC - Port Phy BER Monitor Register Layout,” on page 1406
PPBME	0x5053	v		v		v	v	See Table 1215, “PPBME - Port Phy BER Monitor Register Layout,” on page 1409
PTER	0x5055	v	v		v	v	v	See Table 1076, “PTER- Port Transmit Errors Register Layout,” on page 1270
PEPC	0x5056	v	v		v		v	See Table 1042, “PEPC - Port External Phy Control Register Layout,” on page 1246
PMTPS	0x5060	v	v		v	v	v	See Table 972, “PMTPS - Port Module Type and Physical Settings Register Fields,” on page 1163
PHFR	0x5061	v	v		v	v	v	See Table 1217, “PHFR - Port Halt Fsm Register Layout,” on page 1410
PNLB	0x5062	v	v		v	v	v	See Table 1219, “PNLB - Ports Negotiation Logger Binding Register Layout,” on page 1412
PNLR	0x5063	v	v		v	v	v	See Table 1221, “PNLR - Ports Negotiation Logger Register Layout,” on page 1413
PMPT	0x5064	v	v		v	v	v	See Table 1163, “PMPT - Port Module PRBS Test Register Layout,” on page 1352
PMPD	0x5065	v			v	v	v	See Table 1165, “PMPD - Port Module PRBS Diagnostics Register Layout,” on page 1355
PGUID	0x5066	v			v	v		See Table 1227, “PGUID - Port GUID Register Layout,” on page 1416
PMTM	0x5067	v	v		v	v	v	See Table 1229, “PMTM - Port Module Type Mapping Register Layout,” on page 1418

Table 951 - Ports Register Summary

Register Name	Register ID (hex)	Function			Applicable to	Protocol		Reference
		Query	Write	Event	HCA	InfiniBand	Ethernet	
PLAR	0x5068	v				v	v	See Table 1231, “PLAR- Port Local Allocation Register Layout,” on page 1419

28.19.1 PCAM - Ports Capabilities Mask Register

The PCAM register reports the device's supported port features.

NOTE: For HCA devices, this register is supported only when `HCA_CAP.PCAM == 1`.

Table 952 - PCAM - Ports Capabilities Mask Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								feature_group																access_reg_group								00h
																																04h
port_access_reg_cap_mask																																08h-14h
																																18h-24h
feature_cap_mask																																28h-34h
																																38h-4Ch

Table 953 - PCAM - Ports Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	feature_group	Feature list mask index: 0: enhanced_features Values 1 - 255 are Reserved	Index
	7:0	access_reg_group	Access Register ID groups 0: RegID_0x5000_to_0x507F Values 1-255 are Reserved	Index
08h-14h	128	port_access_reg_cap_mask	Supported port's access register bitmaks. Based on <i>access_reg_group</i> index. When bit is set , The register is supported in the device. When <i>access_reg_group == 0</i> : Bit_0: 0x5000 Bit_1: 0x5001 (PCAP) Bit_2: 0x5002 (PMLP) Bit_3: 0x5003 (PMTU) Bit_4: 0x5004 (PTYs) . . Bit_127: 0x507F (PCAM)	RO

Table 953 - PCAM - Ports Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
28h-34h	128	feature_cap_mask	Supported port's enhanced features. Based on <i>feature_group</i> index. When bit is set, The feature is supported in the device: Bit 0: PPCNT_counter_group_Phy_statistical_counter_group Bit 1 - PPCNT_counter_group_Discard_counter_group Bit 3: PTYS_connector_type_supported - [Internal] Bit 4: RX_buffer_fullness_counters_supported - [Internal] Bit 5: Per_lane_raw_errors_in_statistical_physical_counter_group - [Internal] Bit 6: event_indication_supported_in_PDDR - [Internal] Bit 7: FPGA_port_type_support_in_PPCNT - [Internal] Bit 8: pnat_support_in_PPAD_PAOS_PPTB_and_PBMC - [Internal] Bit 9: PFCC - support of stall mask and global rx,tx mask. Bit 10: PMCR - support of tx_disable override. Bit 11: PPCNT_counter_group - IB Extended port counter group Bit 12: PPCNT_Extended_Ethernet_group - tx packet size histograms Bit 13: PTYS_extended_Ethernet_support - [Internal] Bit 14: PMAOS_reset_toggle_support Bit 15: Set_Module_low_power_supported Bit 16: Module_power_optimization_is_supported_in_PPSC Bit 17: Support_of_IB_force_state_machine_in_PTYS - (an_disable) Bit 18: PPCNT_Extended_Ethernet_group_icrc_counter Bit 19: cm2_en_is_supported_in_PCAP Bit 20: time_tolink_up_is_supported_in_PDDR Bit 21: gearbox_die_num_field_is_supported_in_PMDR Bit 22: force_entropy_entropy_calc_entropy_gre_calc - and capability bits are supported in PCMR Bit 23: FEC_override_support_for_50G_per_lane_is_supported_in_PPLM Bit 24: PTYS_PPCNT_PPconnector_type_supported. - [Internal] Bit 25: PTYS_allow_both_extended_and_legacy_protocol_masks Bit 26: PPCNT_eth_per_tc_cong_group_supported Bit 27: pnat_sup_host - the device supports pnat = 2 (host) when set to '1' Bit 28: ppent_no_buffer_discard_uc - this counter is supported in PPCNT when set to '1' Bit 29: pbmc_port_shared_buffer - supported in PBMC. Bit 30: PLR_max_retry_window_support - PPCNT group 0x22 max retry per window Bit 31: Post BER - PPCNT group 0x16, post-FEC BER supported. Bits 32-39 are Reserved Bit 40: PAOS - support of admin_status opcode 5: "up_lowest_rate" Bit 41: PMAOS - support of "optional_notification" field Bit 42: PRBS_polarity_support - support of polarity bits in PPTT and PPRT Bit 43: PDDR_register_is_supported Bit 44: pfcc_buffer_ownership - PFCC buffer ownership is supported. [Internal] Bit 45: force_down is supported in PAOS reg. Bits 46-127 are Reserved	RO

28.19.2 PCAP - Ports Capabilities Register

The PCAP register reports the ports capabilities of the device. User must make sure the cap mask matches the configuration (e.g. remove HDR from PTYS before removing it from cap mask).

Table 954 - PCAP - Ports Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
cm2_en								local_port																									00h
port_capability_mask[127:96]																															04h		
port_capability_mask[95:64]																															08h		
port_capability_mask[63:32]																															0Ch		
port_capability_mask[31:0]																															10h		

Table 955 - PCAP - Ports Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h	31	cm2_en	port_capability_mask[95:80] is supported if set to '1'.	RW
	23:16	local_port	Local port number	Index
04h	31:0	port_capability_mask[127:96]	Sets the PortInfoCapabilityMask: Specifies the supported capabilities of this node. A bit set to 1 for affirmation of supported capability.	RW
08h	31:0	port_capability_mask[95:64]	port_capability_mask[127:96] is mapped to CapabilityMask in PortInfo MAD.	RW
0Ch	31:0	port_capability_mask[63:32]	port_capability_mask[95:80] is mapped to CapabilityMask2 in PortInfo MAD.	RW
10h	31:0	port_capability_mask[31:0]		RW

28.19.3 PMLP - Ports Module to Local Port Register

The PMLP register configures the assignment of modules to local ports.

Table 956 - PMLP - Ports Module to Local Port Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
rxtx	autosplit							local_port																		width								00h
lane0_module_mapping																															04h			
lane1_module_mapping																															08h			
lane2_module_mapping																															0Ch			
lane3_module_mapping																															10h			
lane4_module_mapping																															14h			
lane5_module_mapping																															18h			
lane6_module_mapping																															1Ch			
lane7_module_mapping																															20h			
																															24h-3Ch			

Table 957 - PMLP - Ports Module to Local Port Register Fields

Offset	Bits	Name	Description	Access
00h	31	rxtx	Use different configuration for RX and TX. If this bit is cleared, the TX value is used for both RX and TX. When set, the RX configuration is taken from the separate field. This is to enable backward compatible implementation.	RW
00h	30	autosplit	AutoSplit configuration: 0 - Auto split flow is disabled. 1 - Allow auto split flow Note: in case of misconfiguration between local ports on the same cluster, the “lower” local port setting will take place.	RW
00h	23:16	local_port	Local port number.	Index
	7:0	width	0: unmap_local_port 1: x1 - lane 0 is used 2: x2 - lanes 0,1 are used 4: x4 - lanes 0,1,2 and 3 are used 8: x8 - lanes 0-7 are used Other - reserved	RW

Table 957 - PMLP - Ports Module to Local Port Register Fields

Offset	Bits	Name	Description	Access
04h	31:0	lane0_module_mapping	Module SerDes for lane <i>Module SerDes for lane <i>Up to 8 SerDeses in a module can be mapped to a local port.	RW
08h	31:0	lane1_module_mapping		
0Ch	31:0	lane2_module_mapping		
10h	31:0	lane3_module_mapping		
14h	31:0	lane4_module_mapping		RW
18h	31:0	lane5_module_mapping		
1Ch	31:0	lane6_module_mapping		
20h	31:0	lane7_module_mapping		

Table 958 - Lane to Module Mapping Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 959 - Lane to Module Mapping Fields

Offset	Bits	Name	Description	Access
00h	27:24	rx_lane	RX lane. When <i>rx_tx</i> field is cleared, this field is ignored and RX lane is equal to TX lane.	RW
	19:16	tx_lane	TX lane. When <i>rx_tx</i> field is cleared, this field is used for RX as well.	RW
	7:0	module	Module number	RW

28.19.4 PMTU - Port MTU Register

The PMTU register configures and reports the port MTU.

Table 960 - PMTU - Port MTU Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																									00h
max_mtu																																04h	
admin_mtu																																08h	
oper_mtu																																0Ch	

Table 961 - PMTU - Port MTU Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
04h	31:16	max_mtu	Maximum MTU supported on the port (Read Only). MTU depends on the port type. When port type (Eth/IB/FC) is configured, the relevant MTU is reported. When the port type is not configured, the minimum between the <i>max_mtu</i> for the different types (Eth/IB/FC) is reported.	RO
08h	31:16	admin_mtu	Administratively configured MTU on the port. Must be smaller or equal to <i>max_mtu</i> .	RW
0Ch	31:16	oper_mtu	Operational MTU. This is the actual MTU configured on the ports. Packets exceeding this size will be dropped. Note: In Ethernet or FC, <i>oper_mtu</i> is equal to <i>admin_mtu</i> . In InfiniBand, the actual operational MTU is reported which may be smaller than <i>admin_mtu</i> .	RO

28.19.5 PTYS - Port Type and Speed Register

The PTYS register configures and reports the port type speed.

Note: Setting the admin fields of this register shall only take effect during link training and negotiation. When set while the link is up, the changes will not take effect unless the port goes back to training and negotiation state (during transition from down to up state). Software can force training by disabling and enabling the port.

Table 962 - PTYS - Port Type and Speed Register Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																		
reserved high	an disable admin	an disable cap				force tx aba param	local_port												port_type														proto_mask	00h																
							an_status			max_port_rate									data_rate_oper																04h															
							ext_eth_proto_capability																															08h												
							eth_proto_capability																															0Ch												
							ib_link_width_capability															ib_proto_capability																10h												
							ext_eth_proto_admin																															14h												
							eth_proto_admin																															18h												
							ib_link_width_admin															ib_proto_admin																1Ch												
							ext_eth_proto_oper																															20h												
							eth_proto_oper																															24h												
							ib_link_width_oper															ib_proto_oper																28h												
																															connector - type	2Ch																		
eth_proto_lp_advertise																															30h																			
																																34-3Ch																		

Table 963 - PTYS - Port Type and Speed Register Fields

Offset	Bits	Name	Description	Access
00h	31	reserved_high reserved	THIS BIT MUST BE TIED TO 1! This is old capability bit that address AN_DISABLE capability with only one enabled protocol.	RO
	30	an_disable_admin	Auto Negotiation disable: 0 - Normal operation 1 - Disable AN. Note: In Ethernet port, when Disabling AN, the “eth_proto_admin” bit mask must comply to single speed rate set. In IB port, when Disabling AN, the “ib_proto_admin” bit mask must comply to single speed rate set. It’s recommended to validate the FEC override bits in PPLM when operating with AN. [Internal] NOTE: if bit 29 (“an_disable_cap”) is clear while bit 31 is ‘1’, AN disable is supported, but “eth_proto_admin” bit mask can be set only with one protocol.	RW
	29	an_disable_cap	Auto Negotiation disable capability: 0 - Device does not support AN disable 1 - Device Supports AN disable	RO
	24	force_tx_aba_param	When AN is disabled, use aba TX set. When set, use ABA parameters on TX when in force operation (AN-disable or parallel detect)	RW
	23:16	local_port	Local port number	Index
	11:8	port_type	Supported only when indicated by PCAM 0: Network_Port 1: Near-End_Port - (For Gearbox - Host side) 2: Internal_IC_Port 3: Far-End_Port - (For Gearbox - Line side) Other values are reserved.	Index
	2:0	proto_mask	Protocol Mask. Indicates which of the protocol data is valid Bit 0: InfiniBand Bit 1: Reserved Bit 2: Ethernet	Index

Table 963 - PTYS - Port Type and Speed Register Fields

Offset	Bits	Name	Description	Access
04h	31:28	an_status	Auto Negotiation status: 0 - Status is unavailable 1 - AN completed successfully 2 - AN performed but failed 3 - AN was not performed, link is up 4 - AN was not performed, link is down	RO
	27:16	max_port_rate	Port maximum data rate in resolution of 1 Gb/s (<i>data_rate_oper</i> * 1 Gb/s) Value 0x0 indicates this field is not supported. [Internal] FW implementation notice - logic should include HDR100 ini value as indication to maximum of 100G even when HDR and 4x are supported.	RO
	15:0	data_rate_oper	Port data rate in resolution of 100 Mb/s (<i>data_rate_oper</i> * 100 Mb/s) Value 0x0 indicates this field is not supported.	RO
08h	31:0	ext_eth_proto_capability	For HCA: See also <i>PCAM.feature_cap_mask</i> bit 13 for Extended Ethernet protocol support. Extended Ethernet port speed/protocols supported (bitmask): Bit 0 - SGMII_100M Bit 1 - 1000BASE-X / SGMII [Internal] Bit 2 - 2.5GBASE-X / 2.5GMII Bit 3 - 5GBASE-R Bit 4 - XFI / XAUI-1 // 10G Bit 5 - XLAUI-4/XLPPI-4 // 40G Bit 6 - 25GAUI-1/ 25GBASE-CR / KR Bit 7 - 50GAUI-2 / LAUI-2/ 50GBASE-CR2/KR2 Bit 8 - 50GAUI-1 / LAUI-1/ 50GBASE-CR / KR Bit 9 - CAUI-4 / 100GBASE-CR4 / KR4 Bit 10 - 100GAUI-2 / 100GBASE-CR2 / KR2 Bit 11 - 100GAUI-1 / 100GBASE-CR / KR Bit 12 - 200GAUI-4 / 200GBASE-CR4/KR4 Bit 13 - 200GAUI-2 / 200GBASE-CR2/KR2 Bit 14 - Reserved Bit 15 - 400GAUI-8 Bit 16 - 400GAUI-4 Other - Reserved	RO

Table 963 - PTYS - Port Type and Speed Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	eth_proto_capability	Ethernet port speed/protocols supported (bitmask) Bit 31 - 50GBase-KR2 Bit 30 - 50GBase-CR2 Bit 29 - 25GBase-SR Bit 28 - 25GBase-KR Bit 27 - 25GBase-CR [Internal] Bit 26 - 10GBase-T [Internal] Bit 25 - 1000Base-T [Internal] Bit 24 - 100Base-TX [Internal] Bit 23 - 100GBase LR4/ER4 Bit 22 - 100GBase KR4 Bit 21 - 100GBase SR4 Bit 20 - 100GBase CR4 Bit 19 - 50GBase-KR4 Bit 18 - 50GBase-SR2 Bit 16 - 40GBase LR4/ER4 Bit 15 - 40GBase SR4 Bit 14 - 10GBase ER/LR Bit 13 - 10GBase SR Bit 12 - 10GBase CR Bit 9 - SGMII_100Base [Internal] Bit 8 - 56GBase *R4 ¹ Bit 7 - 40GBase KR4 Bit 6 - 40GBase CR4 [internal] Bit 5 - 20GBase-KR2 ² Bit 4 - 10GBase KR Bit 3 - 10GBase KX4 Bit 2 - 10GBase-CX4 Bit 1 - 1000Base KX Bit 0 - SGMII	RO
10h	31:16	ib_link_width_capability	ib_link_width <= ib_proto_capability Bit 0 - 1x ³ Bit 1 - 2x Bit 2 - 4x Bit 3 - 8x Bit 4 - 12x	RO
	15:0	ib_proto_capability	InfiniBand port speed supported (bitmask) ib_link_speed <= ib_proto_capability[7:0] Bit 0: SDR ⁴ Bit 1: DDR Bit 2: QDR Bit 3: FDR10 Bit 4: FDR Bit 5: EDR Bit 6: HDR Bit 7: NDR	RO
14h	31:0	ext_eth_proto_admin	Ethernet port extended speed/protocols bitmask NOTE: This field and “ <i>eth_proto_admin</i> ” are mutual exclusive, meaning that only one of the field can be set on write command.	RW

Table 963 - PTYS - Port Type and Speed Register Fields

Offset	Bits	Name	Description	Access
18h	31:0	eth_proto_admin ⁵	Ethernet port speed/protocols bitmask	RW
1Ch	31:16	ib_link_width_admin	InfiniBand port link width bitmask	RW
	15:0	ib_proto_admin ⁶	InfiniBand port speed bitmask	RW
20h	31:0	ext_eth_proto_oper	Ethernet port extended speed/protocols bitmask	RO
24h	31:0	eth_proto_oper	Ethernet port speed/protocols bitmask	RO
28h	31:16	ib_link_width_oper	InfiniBand port link width bitmask	RO
	15:0	ib_proto_oper	InfiniBand port speed bitmask	RO
2Ch	3:0	connector_type	Connector type indication 0: No_connector_or_unknown 1: PORT_NONE - None 2: PORT_TP - Twisted Pair 3: PORT_AUI - AUI 4: PORT_BNC - BNC 5: PORT_MII - MII 6: PORT_FIBRE - FIBRE 7: PORT_DA - Direct Attach Copper 8: PORT_OTHER - Other	RO
30h	31:0	eth_proto_lp_advertise	Ethernet port's link partner advertised protocols. Returns the protocols that were advertised by the link partner during auto-negotiation.	RO

1. [Internal] 56GE can't be set in case of AN disable (i.e. no support of 56GE FORCE mode)
2. [Internal] KR2 has two flavors - HP and Mellanox. This bit should cover both while FW should be able to choose the right protocol based on MLPN and sensing.
3. The InfiniBand specification implies that link width of 1 lane cannot be disabled. Trying to clear this bit will fail the operation.
4. The InfiniBand specification implies that SDR link speed cannot be disabled. Trying to clear this bit will fail the operation.
5. SGMII_100Base can be set only when PTYS.an_disable is set
6. Updating the ib_proto_admin will change the LinkSpeedEnabled reported in the ExtendedPortInfo

28.19.6 PPAD - Port Physical Address Register

The PPAD register is used to query and configure the per port physical MAC address. By default each Ethernet port is assigned with a factory default MAC. The ports' MACs can be configured to different values than the default. PPAD is used for Ethernet ports only.

Table 964 - PPAD - Port Physical Address Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
			single base mac		pnat																											00h
																																04h
																																08h-0Ch

Table 965 - PPAD - Port Physical Address Register Fields

Offset	Bits	Name	Description	Access
00h	28	single_base_mac	0 - base_mac, local_port should be 0 and mac[6:0]and mac [5:0] are reserved in Spectrum-2 and Spectrum-1 accordingly.HW will set incremental 1 - single_mac - mac of the local_port Reserved for SwitchX/-2.	OP
	27:26	pnat	Port number access type. determines the way local_port is interpreted: 0: Local_port_number 1: Label_port_number 3: Out_of_band_or_PCI	Index
	23:16	local_port	Port number Reserved when single_base_mac = 0	Index
	15:0	mac[47:32]	If single_base_mac = 0 - base MAC address, For Spectrum mac[5:0] is reserved.	RW
04h	31:0	mac[31:0]	For Spectrum-2 mac[6:0] is reserved. For HCA mac[7:0] is reserved If single_base_mac = 1 - the per port MAC address	

28.19.7 PAOS - Ports Administrative and Operational Status Register

The PAOS register configures and retrieves the per port administrative and operational status.

Table 966 - PAOS - Ports Administrative and Operational Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
swid								local_port								pnat			admin_status								oper_status				00h	
asc	ec																					fd								e	04h	
																															08h-0Ch	

Table 967 - PAOS - Ports Administrative and Operational Status Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid reserved	For HCA: must always be 0. Switch partition ID with which to associate the port. Switch partitions are numbered from 0 to 7 inclusively. The <i>swid</i> field is only valid when the <i>local_port</i> is the router port. In this case, the <i>swid</i> indicates which of the router ports to configure/query.	Index
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0: Local_port_number 1: Label_port_number 3: Out_of_band_or_PCI	Index
	11:8	admin_status	Port administrative state (the desired state of the interface): 1: up 2: down_by_configuration 3: up_once - if the port goes up and then down, the operational status should go to “down by port failure” and can only go back up upon explicit command 4: disabled_by_system - this mode cannot be set by the software, only by the hardware. 5: up_lowest_rate - when set, if port in down state, there will be an attempt to raise the link in the lowest available rate. [Internal] opcode 5 is used only in NIC, for fast link up drivers after device boot. For most usecase the rate will be SDR, lowest rate support in IB systems. [Internal] Default should come from ini. Up for unmanaged systems and Down for managed [Internal] For InfiniBand ports, this holds the local admin status. SM status does not override this one and should be held in a temporary register. [Internal] if ext_down_admin is in test_mode <i>admin_status</i> must remain down.	RW
	3:0	oper_status	Port operational state: 1: up 2: down 4: down_by_port_failure - (transitioned by the hardware) [Internal] down by port failure state is used to indicate the port was enabled by the admin but moved to down. The device will not let the port up again without software explicit command (UP), i.e <i>admin_status</i> was 4'b0011.	RO

Table 967 - PAOS - Ports Administrative and Operational Status Register Fields

Offset	Bits	Name	Description	Access
04h	31	ase	Admin state update enable. If this bit is set, admin state will be updated based on <i>admin_state</i> field. Only relevant on Set() operations.	WO
	30	ee	Event update enable. If this bit is set, event generation will be updated based on the <i>e</i> field. Only relevant on Set operations. Not Supported for HCA.	WO
	8	fd	Force down. Supported only when indicated in PCAM When set accompanied with <i>admin_status</i> = 2 ('down_by_configuration'), will force link to be down. [Internal] overrides Multi-Host or management requirements.	RW
	1:0	e	Event generation on operational state change: 0: Do_not_generate_event 1: Generate_Event 2: Generate_Single_Event Not Supported for HCA.	RW

28.19.8 PPAOS - Ports Phy Administrative and Operational Status Register

The PPAOS register configures and retrieves the per phy test-mode administrative and operational status.

It can be set to test-mode only when the port is disabled by PAOS. The test mode must be disabled by PPAOS prior to enabling the regular operation of the port using the PAOS.

Table 968 - PPAOS - Port Phy Administrative and Operational Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
swid								local_port												phy_test_- mode_admin				port_type				phy_test_- mode_status				00h	
pdj																													phy_status				04h
																																	08h
																																	0Ch

Table 969 - PPAOS - Port Phy Administrative and Operational Status Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid	For HCA: must always be 0. Switch partition ID with which to associate the port. Switch partitions are numbered from 0 to 7 inclusively. The <i>swid</i> field is only valid when the <i>local_port</i> is the router port. In this case, the <i>swid</i> indicates which of the router ports to configure/query.	Index
	23:16	local_port	Local port number.	Index
	11:8	phy_test_mode_admin	Port extended administrative down status: 0: regular_operation (port down/up according to PAOS) 1: phy_test_mode The phy test mode (1) can be set only when <i>PAOS.admin_status</i> =2 (i.e port configuration is down).	RW
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index
	3:0	phy_test_mode_status	Port extended down status: 0: regular_operation - (port down/up according to PAOS) 1: phy_test_mode <i>phy test mode</i> can be valid only when <i>PAOS.admin_status</i> =2 and <i>PAOS.oper_status</i> =2 (i.e port is down).	RO
04h	31	fpd	Force port down.	RW
04h	3:0	phy_status	rx phy status: 0: rx_phy_down 1: rx_phy_up Note: The phy up indication is according to protocol (up == align_status=true) i.g. In PRBS test mode phy up is when PRB is lock (up== prbs lock)	RO

28.19.9 PMAOS - Ports Module Administrative and Operational Status Register

The PMAOS register configures and retrieves the per module status.

Table 970 - PMAOS - Ports Administrative and Operational Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
1st								module												admin_status								oper_status				00h

Table 970 - PMAOS - Ports Administrative and Operational Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
ase	ce											operation- al_notification								error_type												e	04h
																																	08h
																																	0Ch

Table 971 - PMAOS - Ports Module Administrative and Operational Status Register Fields

Offset	Bits	Name	Description	Access
00h	31	rst	Module Reset toggle NOTE: setting reset while module is plugged-in will result in transition of <i>oper_status</i> to initialization.	OP
	23:16	module	Module number.	Index
	11:8	admin_status	Module administrative state (the desired state of the module): 1: enabled 2: disabled_by_configuration 3: enabled_once - if the module is active and then unplugged, or module experienced an error event, the operational status should go to “disabled” and can only be enabled upon explicit enable command. Note - To disable a module, all ports associated with the port must be disabled first. [Internal] Default should come from ini. Enabled for unmanaged systems and disabled for managed[Internal] For InfiniBand ports, this holds the local admin status. SM status does not override this one and should be held in a temporary register.	RW
	3:0	oper_status	Module state (reserved while <i>admin_status</i> is disabled): 0: initializing ¹ 1: plugged_enabled 2: unplugged 3: module_plugged_with_error - (details in <i>error_type</i>). 4: plugged_disabled	RO

Table 971 - PMAOS - Ports Module Administrative and Operational Status Register Fields

Offset	Bits	Name	Description	Access
04h	31	ase	Admin state update enable. If this bit is set, admin state will be updated based on <i>admin_state</i> field. Only relevant on Set() operations.	WO
	30	ee	Event update enable. If this bit is set, event generation will be updated based on the <i>e</i> field. Only relevant on Set operations.	WO
	19:16	operational_notification	This notification can occur only if module passed initialization process 0x0: No notifications. 0x1: Speed degradation – the module is not enabled in its full speed due to incompatible transceiver/cable Valid only when <i>oper_status</i> = 4'b0001.	RO
	11:8	error_type ²	Module error details: 0x0: Power_Budget_Exceeded 0x1: Long_Range_for_non_MLNX_cable_or_module 0x2: Bus_stuck - (I2C Data or clock shorted) 0x3: bad_or_unsupported_EEPROM 0x4: Enforce_part_number_list 0x5: unsupported_cable 0x6: High_Temperature 0x7: bad_cable - (Module/Cable is shorted) 0x8: PMD_type_is_not_enabled - (see PMTPS). [internal]0x9: Laster_TEC_failure [internal]0xa: High_current [internal]0xb: High_voltage 0xc: pcie_system_power_slot_Exceeded [internal]0xd: High_power [internal]0xe: Module_state_machine_fault Valid only when <i>oper_status</i> = 4'b0011.	RO
	1:0	e	Event Generation on operational state change: 0: Do_not_generate_event 1: Generate_Event 2: Generate_Single_Event	RW

1. When a new cable is plugged or after enable command, the module is being initialized. During module initialization the module can not be accessed. Initialization flow might take a few seconds.
2. In case of more than one error, one of the errors will be reported.

28.19.10[Internal] PMTPS - Port Module Type and Physical Settings Register

The PMTPS register configures and reports the module type

Table 972 - PMTPS - Port Module Type and Physical Settings Register Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 972 - PMTPS - Port Module Type and Physical Settings Register Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																04h
																																08h
																																0Ch
																																10h
																																14h-1Ch

Table 973 - PMTPS - PModule Type and Physical Settings Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
08h	15:0	module_type_admin	Module type enabled (bitmask): Bit 0 - SMF (Single Mode Fiber) Module up to 500m Bit 1 - SMF (Single Mode Fiber) Module between 500m and 2km Bit 2 - SMF (Single Mode Fiber) Module above 2km Bit 3 - MMF (Multi Mode Fiber) Module up to 100m Bit 4 - MMF (Multi Mode Fiber) Module above 100m Bit 5 - AOC / ACC up to 30m Bit 6 - AOC / ACC above 30m Bit 7 - Passive Copper Bit 8 - Twisted Pair Bit 9 - Backplane Set a value of 0xFFFF enables all PMDs type (including of unknowns types)	RW
0Ch	15:0	module_type_connected	Connected and operational PMD type: 0x0 - Unknown / Not Connected 0x1 - SMF (Single Mode Fiber) Module up to 500m 0x2 - SMF (Single Mode Fiber) Module up to 2km 0x4 - SMF (Single Mode Fiber) Module above 2km 0x8 - MMF (Multi Mode Fiber) Module up to 100m 0x10 - MMF (Multi Mode Fiber) Module above 100m 0x20 - AOC / ACC up to 30m 0x40 - AOC / ACC up to 30m 0x80 - Passive copper 0x100 - Twisted Pair 0x200 - Backplane	RO

Table 973 - PMTPS - PModule Type and Physical Settings Register Fields

Offset	Bits	Name	Description	Access
10h	31:0	eth_module_c2m	Supported Module electrical interfaces: Bit 0 - SGMII_100M Bit 1 - 1000BASE-X / SGMII [Internal] Bit 2 - 2.5GBASE-X / 2.5GMII Bit 3 - 5GBASE-R Bit 4 - XFI / SFI // 10G Bit 5 - XLAUI-4/XLPPI-4 // 40G Bit 6 - 25GAUI-1/ 25GBASE-CR / KR Bit 7 - 50GAUI-2 / LAUI-2/ 50GBASE-CR2/KR2 Bit 8 - 50GAUI-1 / LAUI-1/ 50GBASE-CR / KR Bit 9 - CAUI-4 / 100GBASE-CR4 / KR4 Bit 10 - 100GAUI-2 / 100GBASE-CR2 / KR2 Bit 11 - Reserved Bit 12 - 200GAUI-4 / 200GBASE-CR4/KR4 Bit 13 - Reserved Bit 14 - Reserved Bit 15 - 400GAUI-8 Other - Reserved	RO

28.19.11PFCC - Ports Flow Control Configuration Register

The PFCC register configures and retrieves the per port flow control configuration.

Table 974 - PFCC - Ports Flow Control Configuration Register Layout

Offset	00h	04h	08h	0Ch	10h	14h	18h	1Ch		
0	shl opr	prio_mask_rx	cbfix	cbfix						
1	shl cap									
2	critical stall mask									
3	minor stall mask									
4	ppan mask n									
5										
6										
7										
8			fcix disabled							
9										
10										
11										
12										
13										
14	pnat									
15										
16	local_port									
17										
18										
19										
20										
21										
22										
23										
24	set_buf_ownership									
25	buf_ownership									
26										
27										
28	cap remote admin									
29	cap local admin									
30	decbx_operation_type	ppan	ppix	pprx						
31			ppix mask n apix	pprx mask n aprx						

Note: Only a single flow control mechanism can be used on a specific port (for both RX and TX). [Internal] In order to configure a port to have no flow control, all types of flow control should be explicitly reset.

Note: Setting pptx / pprx when the link is up with ppan = 0 takes effect immediately. Setting pptx / pprx with ppan = 1 only take effect during the link training and negotiation. Soft-

ware can force training by disabling and enabling the port. Setting `pfctx` / `pfcrx` when the link is up takes effect immediately.

Table 975 - PFCC - Ports Flow Control Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	dcxb_operation_type <i>reserved</i>	The Operation type, valid only on HCAs. 0x0: local_operative_parameters 0x1: local_admin_parameters - supported only when <i>cap_local_admin==1</i> . 0x2: remote_admin_parameters - supported only when <i>cap_remote_admin==1</i> . Only <i>pfctx</i> and <i>pfcrx</i> are supported on remote device	Index
	29	cap_local_admin	Indicates if operation of local admin parameters (<i>dcxb_operation_type</i> 0x1) is supported. <i>[Internal] In all Switches the value of this field shall be 0</i>	RO
	28	cap_remote_admin	Indicates if operation of remote admin parameters (<i>dcxb_operation_type</i> 0x2) is supported. <i>[Internal] In all Switches the value of this field shall be 0</i>	RO
	27:26	buf_ownership	Supported by HCA only, see PCAM.feature_cap_mask bit 44. Current status of buffer ownership 0 - unknown 1 - device owned 2 - software owned	RO
	25:24	set_buf_ownership	Supported by HCA only, see PCAM.feature_cap_mask bit 44. Buffer ownership configuration. when this bits are set to a value other than 0, device will ignore all other fields pfcc register 0 - unknown 1 - device owned, all buffer configuration will return to its defaults. 2 - Reserved 3 - Reserved	WO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number <i>[Internal] 3 - Out of band / PCI</i>	Index
	4	ppan_mask_n	Valid only on HCAs. Mask bit (negative polarity) indicating if <i>ppan</i> should be updated. If set to '0', <i>ppan</i> will be updated	WO
	3	minor_stall_mask	Valid only on HCAs. Mask bit indicating if <i>device_stall_minor_watermark</i> should be updated. If set to '1', <i>device_stall_minor_watermark</i> will be updated	WO

Table 975 - PFCC - Ports Flow Control Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	2	critical_stall_mask	Valid only on HCAs. Mask bit indicating if <i>device_stall_critical_watermark</i> should be updated. If set to '1', <i>device_stall_critical_watermark</i> will be updated	WO
	1	shl_cap	Send to higher layers capabilities: 0 - No capability of sending Pause and PFC to higher layers 1 - Device has capability of send Pause and PFC to higher layers NOTE: This relates to IEEE 802.3 PAUSE and IEEE 802.3 PFC.	RO
	0	shl_opr	Send to higher layers operation: 0 - Pause and PFC are handled by the port (default) 1 - Pause and PFC are handled by the port and are also been send to higher layers. Only valid if <i>shl_cap</i> = 1.	RW
04h	31:28	ppan <i>reserved</i>	Pause policy auto negotiation 0: disabled - generate and ignore pause frames based on pptx, pprx 1: enabled - when auto-negotiation is performed, set the pause policy based on the auto-negotiation resolution. Note - The auto-negotiation advertisement is set according to pptx, pprx. Note - When PFC is set on the TX / RX ppan must be set to 0.	RW
	23:16	prio_mask_tx	Bit per prio indicating if TX flow control policy should be updated based on bit <i>pfctx</i> and <i>cbctx</i> .	WO
	7:0	prio_mask_rx	Bit per prio indicating if RX flow control policy should be updated based on bit <i>pfcrx</i> and <i>cbcrx</i> .	WO

Table 975 - PFCC - Ports Flow Control Configuration Register Fields

Offset	Bits	Name	Description	Access
08h	31	pptx	Admin pause policy on TX (see also <i>pfctx</i>): 0 - never generate pause frames (default) 1 - generate pause frames according to RX buffer threshold	RW
	30	aptx	Active (operational) pause policy on TX 0 - do not generate pause frames 1 - generate pause frames according to RX buffer threshold	RO
	29	pptx_mask_n	Valid only on HCAs. Mask bit (negative polarity) indicating if <i>pptx</i> should be updated. If set to '0', <i>pptx</i> will be updated	WO
	23:16	pfctx	Priority based flow control policy on TX[7:0]. Per priority bit mask: 0 - never generate pause frames on the specified priority (default) 1 - generate pause frames according to RX buffer threshold on the specified priority <i>pfctx</i> , <i>pptx</i> and <i>cbftx</i> must be mutually exclusive (for example, only one of them at most can be set).	RW
	8	fctx_disabled	Valid only on HCAs. The bit is set if the device has passed the <i>device_stall_critical_watermark</i> (for global flow control or any priority when using priority flow control) and has become stalled. When <i>fctx_disabled</i> is set, the device will not send flow control and priority flow control (PFC) packets. ~~[Internal] When the mechanism works on all priorities, blocking traffic, all priorities with PFC enable (or all priorities in case of global pause) will be set.~~	RO
	7:0	cbftx	Credit based flow control policy on TX[7:0]. Per TClass bit mask: 0 - never generate flow control credit frames on the specified TClass (default) 1 - generate flow control credit frames according to RX buffer state on the specified TClass <i>pfctx</i> , <i>pptx</i> and <i>cbftx</i> must be mutually exclusive (i.e. only one of them at most can be set).	RW

Table 975 - PFCC - Ports Flow Control Configuration Register Fields

Offset	Bits	Name	Description	Access
0Ch	31	pprx	Admin pause policy on RX (see also <i>pfcrx</i>): 0 - ignore received pause frames (default) 1 - respect received pause frames	RW
	30	aprx	Active (operational) pause policy on RX 0 - ignore received pause frames 1 - respect received pause frames	RO
	29	pprx_mask_n	Valid only on HCAs. Mask bit (negative polarity) indicating if <i>pprx</i> should be updated. If set to '0', <i>pprx</i> will be updated	WO
	23:16	pfcrx	Priority based flow control policy on RX[7:0]. Per priority bit mask: 0 - ignore incoming pause frames on the specified priority (default) 1 - respect incoming pause frames on the specified priority	RW
	7:0	cbfrx	Credit based flow control policy in RX[7:0]. Per TClass bit mask: 0 - ignore incoming flow control credit frames on the specified TClass (default) 1 - respect flow control credit frames according to RX buffer state on the specified TClass	RW
10h	31:16	device_stall_minor_watermark	Valid only on HCAs (When ~~rx_activity~~ <i>health_mon_rx_activity</i> is set). The maximum period for a single received packet processing, if the packet was not processed during this time, the device will increase the <i>device_stall_minor_watermark_cnt (PPCNT)</i> . Value given in mSec, The maximum period is 8 sec. The special value of 0, indicates that the <i>device_stall_minor_watermark</i> is inactive. Range: 0x0050 - 0x1F40 [internal] granularity is 50 ms see also <i>critical_stall_mask</i>	RW
	15:0	device_stall_critical_watermark	Valid only on HCAs (When ~~rx_activity~~ <i>health_mon_rx_activity</i> is set). The maximum period for a single received packet processing, if the packet was not processed during this time, the device will be declared as stalled and will increase the <i>device_stall_critical_watermark_cnt (PPCNT)</i> counter. Value given in mSec, The maximum period is 8 sec. The special value of 0, indicates that the <i>device_stall_critical_watermark</i> is inactive. Range: 0x0050 - 0x1F40 [internal] granularity is 50 ms see also <i>critical_stall_mask</i>	RW

<TBD> Need to define the register behavior for IB.

28.19.12PPCNT - Ports Performance Counters Register

The PPCNT register retrieves per port performance counters.

Table 976 - PPCNT - Ports Performance Counters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
swid								local_port								pnat			port_type						grp					00h			
clr																												prio_tc					04h
counter_set																															08h-FCh		

<TBD> Do we need sub port???

Table 977 - PPCNT - Ports Performance Counters Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid reserved	For HCA: must be always 0. Switch partition ID to associate port with. Switch partitions are numbered from 0 to 7 inclusively. Switch partition 254 indicates stacking ports. Switch partition 255 indicates all switch partitions. Only valid on Set() operation with local_port=255.	Index
	23:16	local_port	Local port number. 255 indicates all ports on the device, and is only allowed for Set() operation.	Index
	15:14	pnat	Port number access type: 0: Local_port_number 1: IB_port_number 2: host_port_number 3: OOB_port_number - Internal Note - pnat=2 is supported for “Per Traffic Class Congestion Counters” and “Per Traffic Class Counters” groups.	Index
	11:8	port_type	Supported only when indicated by PCAM on FPGA based NICs: 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 -Main USR - groups 0x12 and 0x14 5 - Tile USR - groups 0x12 and 0x14	Index

Table 977 - PPCNT - Ports Performance Counters Register Fields

Offset	Bits	Name	Description	Access
00h	5:0	grp	Performance counter group. Group 63 indicates all groups (include all per priority/TC/Receive Buffer counters). Only valid on Set() operation with clr bit set. Note: Groups that are valid for a OOB are only 0x10 and 0x19 Note: Group 0x24 can be accessed only when port_type_index=4 (main USR) or when port_type_index=5 (Tile USR) 0x0: IEEE_802_3_Counters 0x1: RFC_2863_Counters 0x2: RFC_2819_Counters 0x3: RFC_3635_Counters 0x5: Ethernet_Extended_Counters 0x6: Ethernet_Discard_Counters 0x8: Link_Level_Retransmission_Counters - [Internal] 0x10: Per_Priority_Counters - [Internal]: Some of the fields in this group are valid for OOB ports 0x11: Per_Traffic_Class_Counters - [Internal]: Some of the fields in this group are valid for OOB ports 0x12: Physical_Layer_Counters 0x13: Per_Traffic_Class_Congestion_Counters 0x14: ZL_FEC_Counters - [Internal] 0x15: Per_Receive_Buffer_Counters - [Internal] 0x16: Physical_Layer_Statistical_Counters 0x19: Out_of_Band_Port_Counters - Note: This group is valid only for OOB ports - [Internal] 0x20: InfiniBand_Port_Counters 0x21: InfiniBand_Extended_Port_Counters 0x22: PLR_counters_group 0x23: RS_FEC_Histogram_group - [Internal] 0x24: USR_XSR_Physical_layer_counters_group - [Internal] TBD Group 0x13 (Per Traffic Class Congestion Counters) is valid only for Switches	Index
04h	31	clr	Clear counters. Setting the clr bit will reset the counter value for all counters in the counter group. This bit can be set for both Set() and Get() operation. NOTE: when clr bit is set, all port_type are cleared.	OP
	4:0	prio_tc	Priority index for per priority counter sets, valid values: 0-7 For Spectrum-2 and on at tx side range can be 0 .. cap_max_tclass_data-1, see QGCR.tx_cnt_tclass Traffic class index for per traffic class counter set, valid values: For Switches, valid values: 0 .. cap_max_tclass_data-1 For HCA, valid values: 0.. HCA_CAP.max_tc For "Receive Buffer Group" index per receive buffer counter set: For Switches only, valid values: 0 .. cap_max_pg_buffers -1 Otherwise must be 0.	Index

Table 977 - PPCNT - Ports Performance Counters Register Fields

Offset	Bits	Name	Description	Access
08h-FCh	1984	counter_set	Counter set as described in See Table 978, “Ethernet IEEE 802.3 Counters Group Data Layout,” on page 1173 See Table 980, “Ethernet RFC 2863 Counter Group Data Layout,” on page 1177 See Table 982, “Ethernet RFC 2819 Counter Group Data Layout,” on page 1181 See Table 984, “Ethernet RFC 3635 Counter Group Data Layout,” on page 1185 See Table 986, “Ethernet Extended Counter Group Data Layout,” on page 1188 See Table 989, “Ethernet Discard Counter Group Fields,” on page 1193 See Table 1000, “Link Level Retransmission Counter Group Data Layout,” on page 1203 See Table 990, “Ethernet Per Priority Group Data Layout,” on page 1195 See Table 994, “Ethernet Per Traffic Class Group data layout,” on page 1201 See Table 1006, “Physical Layer Counters Data Layout,” on page 1210 See Table 996, “Ethernet Per Traffic Class Congestion Group data layout,” on page 1202 [Internal] See Table 1016, “ZL FEC Counters Data Layout,” on page 1223 See Table 998, “Ethernet Per Receive Buffer Group data layout,” on page 1203 See Table 1010, “Physical Layer Statistical Counters Data Layout,” on page 1217 See Table 992, “Ethernet OOB Port Counters Data Layout,” on page 1198 - [Internal] See Table 1002, “InfiniBand PortCounters Attribute Group Data Layout,” on page 1204 See Table 1004, “InfiniBand Extended PortCounters Attribute Group Data Layout,” on page 1206 See Table 1012, “PLR Counters Data Layout,” on page 1221 - [Internal] See Table 1014, “RS-Histograms Data Layout,” on page 1223 - [Internal] See Table 1008, “USR_XSR Physical Layer Counters Data Layout,” on page 1214	RO

28.19.12.1 Ethernet IEEE 802.3 Counters

Table 978 - Ethernet IEEE 802.3 Counters Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
a_frames_transmitted_ok_high																															00h	
a_frames_transmitted_ok_low																															04h	
a_frames_received_ok_high																															08h	
a_frames_received_ok_low																															0Ch	
a_frame_check_sequence_errors_high																															10h	
a_frame_check_sequence_errors_low																															14h	

Table 978 - Ethernet IEEE 802.3 Counters Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
a_alignment_errors_high																															18h	
a_alignment_errors_low																															1Ch	
a_octets_transmitted_ok_high																															20h	
a_octets_transmitted_ok_low																															24h	
a_octets_received_ok_high																															28h	
a_octets_received_ok_low																															2Ch	
a_multicast_frames_xmitted_ok_high																															30h	
a_multicast_frames_xmitted_ok_low																															34h	
a_broadcast_frames_xmitted_ok_high																															38h	
a_broadcast_frames_xmitted_ok_low																															3Ch	
a_multicast_frames_received_ok_high																															40h	
a_multicast_frames_received_ok_low																															44h	
a_broadcast_frames_received_ok_high																															48h	
a_broadcast_frames_received_ok_low																															4Ch	
a_in_range_length_errors_high																															50h	
a_in_range_length_errors_low																															54h	
a_out_of_range_length_field_high																															58h	
a_out_of_range_length_field_low																															5Ch	
a_frame_too_long_errors_high																															60h	
a_frame_too_long_errors_low																															64h	
a_symbol_error_during_carrier_high																															68h	
a_symbol_error_during_carrier_low																															6Ch	
a_mac_control_frames_transmitted_high																															70h	
a_mac_control_frames_transmitted_low																															74h	
a_mac_control_frames_received_high																															78h	
a_mac_control_frames_received_low																															7Ch	
a_unsupported_opcodes_received_high																															80h	
a_unsupported_opcodes_received_low																															84h	
a_pause_mac_ctrl_frames_received_high																															88h	
a_pause_mac_ctrl_frames_received_low																															8Ch	

Table 978 - Ethernet IEEE 802.3 Counters Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
a_pause_mac_ctrl_frames_transmitted_high																																90h
a_pause_mac_ctrl_frames_transmitted_low																																94h
																																98-F4h

Table 979 - Ethernet IEEE 802.3 Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	a_frames_transmitted_ok_high	A count of frames that are successfully transmitted. [Internal] Includes control (pause) frames. FW: port_xmit_pkts + tpause_global HW bug: counts also bad packets (EBP) in cut-through	RO
04h	31:0	a_frames_transmitted_ok_low		RO
08h	31:0	a_frames_received_ok_high	A count of frames that are successfully received. This does not include frames received with frame-too-long, frame check sequence (FCS), length or alignment errors, or frames lost due to other MAC errors. [Internal] Includes frames discarded due to buffer overflow, according to spec. Switch FW: frame_received_ok	RO
0Ch	31:0	a_frames_received_ok_low		RO
10h	31:0	a_frame_check_sequence_errors_high	A count of receive frames that are an integral number of octets in length and do not pass the FCS check. This does not include frames received with frame-too-long, or frame-too-short (frame fragment) errors. [Internal] also includes phy errors as described in RFC3635 dot3StatsFCSErrors FW: rrcr + received_symbol_errors	RO
14h	31:0	a_frame_check_sequence_errors_low		RO
18h	31:0	a_alignment_errors_high	A count of frames that are not an integral number of octets in length and do not pass the FCS check. [Internal] Not Implemented (always return zero)	RO
1Ch	31:0	a_alignment_errors_low		RO
20h	31:0	a_octets_transmitted_ok_high	A count of data and padding octets of frames that are successfully transmitted. [Internal] Includes control (pause) frames. FW: port_xmit_data + 64*tpause_global Same issue as “a_frames_transmitted_ok_high” with EBP packets.	RO
24h	31:0	a_octets_transmitted_ok_low		RO
28h	31:0	a_octets_received_ok_high	A count of data and padding octets in frames that are successfully received. This does not include octets in frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to other MAC errors. [Internal] Includes discard due to buffer overflow. Includes control (pause) frames. Counts also Ethernet header and FCS. FW: octets_received_ok_global	RO
2Ch	31:0	a_octets_received_ok_low		RO

Table 979 - Ethernet IEEE 802.3 Counter Group Fields

Offset	Bits	Name	Description	Access
30h	31:0	a_multicast_frames_xmitted_ok_high	A count of frames that are successfully transmitted to a group destination address other than broadcast. [Internal] Switch FW: port_multicast_xmit_pkts Doesn't include Pause and discarded packets (counts only packet arriving to LLU). TBD - do we need to count pause frames? Same issue as "a_frames_transmitted_ok_high" with EBP packets.	RO
34h	31:0	a_multicast_frames_xmitted_ok_low		RO
38h	31:0	a_broadcast_frames_xmitted_ok_high	A count of the frames that were successfully transmitted to the broadcast address. Frames transmitted to multicast addresses are not broadcast frames and are excluded. FW: port_broadcast_xmit_pkts Same issue as "a_frames_transmitted_ok_high" with EBP packets.	RO
3Ch	31:0	a_broadcast_frames_xmitted_ok_low		RO
40h	31:0	a_multicast_frames_received_ok_high	A count of frames that are successfully received and directed to an active nonbroadcast group address. This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. [Internal] Doesn't count packets which are discarded by buffer although by 802.3 spec should count also packets discarded by buffer. Doesn't include Pause. [Internal] Switch FW: in_mcast_pkts. This HW counter also counts for <i>ifInMulticastPkts</i> .	RO
44h	31:0	a_multicast_frames_received_ok_low		RO
48h	31:0	a_broadcast_frames_received_ok_high	A count of the frames that were successfully transmitted to the broadcast address. Frames transmitted to multicast addresses are not broadcast frames and are excluded. FW: in_bcast_pkts same issue as multicast	RO
4Ch	31:0	a_broadcast_frames_received_ok_low		RO
50h	31:0	a_in_range_length_errors_high	A count of frames with a length/type field value between the minimum unpadded MAC client data size and the maximum allowed MAC client data size, inclusive, that does not match the number of MAC client data octets received. The counter also increments for frames whose length/type field value is less than the minimum allowed unpadded MAC client data size and the number of MAC client data octets received is greater than the minimum unpadded MAC client data size. FW: in_range_length_error	RO
54h	31:0	a_in_range_length_errors_low		RO
58h	31:0	a_out_of_range_length_field_high	A count of frames with a length field value greater than the maximum allowed LLC data size. FW: out_of_range_length_field	RO
5Ch	31:0	a_out_of_range_length_field_low		RO
60h	31:0	a_frame_too_long_errors_high	A count of frames received that exceed the maximum permitted frame size by IEEE 802.3 (MTU size). FW: frame_too_long_errors_mtu	RO
64h	31:0	a_frame_too_long_errors_low		RO

Table 979 - Ethernet IEEE 802.3 Counter Group Fields

Offset	Bits	Name	Description	Access
68h	31:0	a_symbol_error_during_carrier_high	For full duplex operation at 1000 Mb/s, it is a count of the number of times the receiving media is non-idle (a carrier event) for a period of time equal to or greater than minFrameSize, and during which there was at least one occurrence of an event that causes the PHY to indicate “Data reception error”. For operation at 10 Gb/s, 40 Gb/s, and 100 Gb/s, it is a count of the number of times the receiving media is non-idle for a period of time equal to or greater than minFrameSize, and during which there was at least one occurrence of an event that causes the PHY to indicate Error. [Internal] return the number of coding errors during packet FW BUG: Always return 0, Switch CCB approved at March 2016 for FW fix. Switch FW: rceivied_symol_error	RO
6Ch	31:0	a_symbol_error_during_carrier_low		RO
70h	31:0	a_mac_control_frames_transmitted_high	A count of MAC Control frames passed to the MAC sublayer for transmission. FW: tpause_global This counts Global Pause and PFC and not include SPF	RO
74h	31:0	a_mac_control_frames_transmitted_low		RO
78h	31:0	a_mac_control_frames_received_high	A count of MAC Control frames passed by the MAC sublayer to the MAC Control sublayer. FW: rpause_global+rcntl This counts Global Pause and PFC and not include SPF	RO
7Ch	31:0	a_mac_control_frames_received_low		RO
80h	31:0	a_unsupported_opcodes_received_high	A count of MAC Control frames received that contain an opcode that is not supported by the device. FW: rcntl This counts Global Pause and PFC and not include SPF	RO
84h	31:0	a_unsupported_opcodes_received_low		RO
88h	31:0	a_pause_mac_ctrl_frames_received_high	A count of MAC PAUSE frames passed by the MAC sublayer to the MAC Control sublayer. FW: rpause_global This counts Global Pause and PFC and not include SPF	RO
8Ch	31:0	a_pause_mac_ctrl_frames_received_low		RO
90h	31:0	a_pause_mac_ctrl_frames_transmitted_high	A count of PAUSE frames passed to the MAC sublayer for transmission. FW: tpause_global This counts Global Pause and PFC and not include SPF	RO
94h	31:0	a_pause_mac_ctrl_frames_transmitted_low		RO

28.19.12.2 Ethernet RFC 2863 Counters

Table 980 - Ethernet RFC 2863 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
if_in_octets_high																															00h	
if_in_octets_low																															04h	

Table 980 - Ethernet RFC 2863 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
if_in_ucast_pkts_high																															08h	
if_in_ucast_pkts_low																															0Ch	
if_in_discards_high																															10h	
if_in_discards_low																															14h	
if_in_errors_high																															18h	
if_in_errors_low																															1Ch	
if_in_unknown_protos_high																															20h	
if_in_unknown_protos_low																															24h	
if_out_octets_high																															28h	
if_out_octets_low																															2Ch	
if_out_ucast_pkts_high																															30h	
if_out_ucast_pkts_low																															34h	
if_out_discards_high																															38h	
if_out_discards_low																															3Ch	
if_out_errors_high																															40h	
if_out_errors_low																															44h	
if_in_multicast_pkts_high																															48h	
if_in_multicast_pkts_low																															4Ch	
if_in_broadcast_pkts_high																															50h	
if_in_broadcast_pkts_low																															54h	
if_out_multicast_pkts_high																															58h	
if_out_multicast_pkts_low																															5Ch	
if_out_broadcast_pkts_high																															60h	
if_out_broadcast_pkts_low																															64h	
																															68-FAh	

Table 981 - Ethernet RFC 2863 Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	if_in_octets_high	The total number of octets received, including framing characters. Including MAC control frames. <i>[Internal] Includes discard due to buffer overflow. Includes control (pause) frames. Counts also Ethernet header and FCS. FW: octets_received_ok_global</i>	RO
04h	31:0	if_in_octets_low		RO
08h	31:0	if_in_ucast_pkts_high	The number of packets successfully received, which were not addressed to a multicast or broadcast MAC address. <i>FW: in_ucast_pkts doesn't count packets which were discarded by buffer.</i>	RO
0Ch	31:0	if_in_ucast_pkts_low		RO
10h	31:0	if_in_discards_high	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. <i>FW: in_discards_global</i> <i>HW :</i> 1. Headroom full: mtu fragment prevention check For lossless: headroom is too small (by RTT) For lossy: headroom is too small (< 2MTU) or pipeline latency 2. discard due to force overflow configuration - FW closes ingress 3. early drop mechanism - TBD not in POR today 4. ptb protection for shared buffer - packets drop because of shared buffer event (not a continues condition)	RO
14h	31:0	if_in_discards_low		RO
18h	31:0	if_in_errors_high	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. (FCS, Phy, over/user size and length). <i>Switch FW: received_symbols_errors+rcrc+rx_stats_under-size_pkts+frames_too_long_errors_mtu+in_range_length_error+out_of_range_length</i>	RO
1Ch	31:0	if_in_errors_low		RO
20h	31:0	if_in_unknown_protos_high	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol. <i>[Internal] Always return zero</i>	RO
24h	31:0	if_in_unknown_protos_low		RO
28h	31:0	if_out_octets_high	The total number of octets transmitted out of the interface, including framing characters. <i>[Internal] Includes control (pause) frames. FW: port_xmit_data + 64*tpause_global</i>	RO
2Ch	31:0	if_out_octets_low		RO
30h	31:0	if_out_ucast_pkts_high	The total number of packets that higher-level protocols requested be transmitted and were not addressed to a multicast or broadcast MAC address, including those that were discarded or not sent. <i>[Internal] Not including Egress policer</i> <i>Switch FW: port_unicast_xmit_pkts</i>	RO
34h	31:0	if_out_ucast_pkts_low		RO

Table 981 - Ethernet RFC 2863 Counter Group Fields

Offset	Bits	Name	Description	Access
38h	31:0	if_out_discards_high	The number of outbound packets which were chosen to be discarded, even though no errors had been detected to prevent their being transmitted. Switch FW: SLL, HLL, WRED, tail-drop NIC FW: Link state, HOQ, STALL, Config (NIC only) TBD: 1) egress policer currently NO 2) link down - FW supports but it will not rise due to FDB.	RO
3Ch	31:0	if_out_discards_low		RO
40h	31:0	if_out_errors_high	The number of outbound packets that could not be transmitted because of errors. Switch FW: discard_mtu HW: <i>tq_general</i> . In Spectrum <i>tq_general</i> is counting only MTU discards	RO
44h	31:0	if_out_errors_low		RO
48h	31:0	if_in_multicast_pkts_high	The number of packets successfully received, which were addressed to a multicast MAC address. [Internal] By spec doesn't count also packets discarded by buffer - TBD check our configuration. [Internal] Switch FW: in_mcast_pkts. Doesn't include Pause This HW counter also counts for <i>a_multicast</i> which is not count also TBD what is configured for in_mcast_pkts	RO
4Ch	31:0	if_in_multicast_pkts_low		RO
50h	31:0	if_in_broadcast_pkts_high	The number of packets successfully received, which were addressed to a broadcast MAC address. FW: in_bcast_pkts same issue as multicast	RO
54h	31:0	if_in_broadcast_pkts_low		RO
58h	31:0	if_out_multicast_pkts_high	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast MAC address, including those that were discarded or not sent. [Internal] Switch FW: port_multicast_xmit_pkts [Internal] Doesn't include Pause and discarded packets (counts only packet arriving to LLU) [Internal] Although discarded packets should be counted they are not.	RO
5Ch	31:0	if_out_multicast_pkts_low		RO
60h	31:0	if_out_broadcast_pkts_high	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a broadcast MAC address, including those that were discarded or not sent. [Internal] Switch FW: port_broadcast_xmit_pkts [Internal] Doesn't include discarded packets (counts only packet arriving to LLU) [Internal] Although discarded packets should be counted they are not.	RO
64h	31:0	if_out_broadcast_pkts_low		RO

28.19.12.3 Ethernet RFC 2819 Counters

Table 982 - Ethernet RFC 2819 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ether_stats_drop_events_high																															00h	
ether_stats_drop_events_low																															04h	
ether_stats_octets_high																															08h	
ether_stats_octets_low																															0Ch	
ether_stats_pkts_high																															10h	
ether_stats_pkts_low																															14h	
ether_stats_broadcast_pkts_high																															18h	
ether_stats_broadcast_pkts_low																															1Ch	
ether_stats_multicast_pkts_high																															20h	
ether_stats_multicast_pkts_low																															24h	
ether_stats_crc_align_errors_high																															28h	
ether_stats_crc_align_errors_low																															2Ch	
ether_stats_undersize_pkts_high																															30h	
ether_stats_undersize_pkts_low																															34h	
ether_stats_oversize_pkts_high																															38h	
ether_stats_oversize_pkts_low																															3Ch	
ether_stats_fragments_high																															40h	
ether_stats_fragments_low																															44h	
ether_stats_jabbers_high																															48h	
ether_stats_jabbers_low																															4Ch	
ether_stats_collisions_high																															50h	
ether_stats_collisions_low																															54h	
ether_stats_pkts64octets_high																															58h	
ether_stats_pkts64octets_low																															5Ch	
ether_stats_pkts65to127octets_high																															60h	
ether_stats_pkts65to127octets_low																															64h	
ether_stats_pkts128to255octets_high																															68h	
ether_stats_pkts128to255octets_low																															6Ch	
ether_stats_pkts256to511octets_high																															70h	

Table 982 - Ethernet RFC 2819 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ether_stats_pkts256to511octets_low																															74h	
ether_stats_pkts512to1023octets_high																															78h	
ether_stats_pkts512to1023octets_low																															7Ch	
ether_stats_pkts1024to1518octets_high																															80h	
ether_stats_pkts1024to1518octets_low																															84h	
ether_stats_pkts1519to2047octets_high																															88h	
ether_stats_pkts1519to2047octets_low																															8Ch	
ether_stats_pkts2048to4095octets_high																															90h	
ether_stats_pkts2048to4095octets_low																															94h	
ether_stats_pkts4096to8191octets_high																															98h	
ether_stats_pkts4096to8191octets_low																															9Ch	
ether_stats_pkts8192to10239octets_high																															A0h	
ether_stats_pkts8192to10239octets_low																															A4h	
																															A8-F4h	

Table 983 - Ethernet RFC 2819 Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	ether_stats_drop_events_high	The total number of events in which packets were dropped by the probe due to lack of resources. Switch FW : in_discards_global Switch HW : 1. discard due to force overflow configuration - FW closes ingress 2. early drop mechanism - TBD not in POR today 3. mtu fragment prevention check - TBD if implemented 4. ptb protection for shared buffer - packets drop because of shared buffer event (not a continues condition)	RO
04h	31:0	ether_stats_drop_events_low		RO
08h	31:0	ether_stats_octets_high	The total number of octets of data (including those in bad packets) received (excluding framing bits but including FCS octets). [Internal] Includes all L2 discards. Includes control (pause) frames - TBD. Switch FW: rx_stats_octets_global	RO
0Ch	31:0	ether_stats_octets_low		RO

Table 983 - Ethernet RFC 2819 Counter Group Fields

Offset	Bits	Name	Description	Access
10h	31:0	ether_stats_pkts_high	The total number of packets (including bad packets, broadcast packets, and multicast packets) received. [Internal] Includes all L2 discards. Includes control (pause) frames - TBD. Switch FW: rx_stats_pkts_global	RO
14h	31:0	ether_stats_pkts_low		RO
18h	31:0	ether_stats_broadcast_pkts_high	The total number of good packets received that were directed to the broadcast address. Note: This does not include multicast packets. Switch FW: in_bcast_pkts	RO
1Ch	31:0	ether_stats_broadcast_pkts_low		RO
20h	31:0	ether_stats_multicast_pkts_high	The total number of good packets received that were directed to a multicast MAC address. Note: This number does not include packets directed to the broadcast address. Switch FW: in_mcast_pkts	RO
24h	31:0	ether_stats_multicast_pkts_low		RO
28h	31:0	ether_stats_crc_align_errors_high	The total number of packets received that had a length (excluding framing bits, but including FCS octets) of between 64 and MTU octets, inclusive, but had either a bad frame check sequence (FCS) with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Switch FW: rrcrc + received_symbol_errors	RO
2Ch	31:0	ether_stats_crc_align_errors_low		RO
30h	31:0	ether_stats_undersize_pkts_high	The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. Switch FW: rx_stats_undersize_pkts	RO
34h	31:0	ether_stats_undersize_pkts_low		RO
38h	31:0	ether_stats_oversize_pkts_high	The total number of packets received that were longer than MTU octets (excluding framing bits, but including FCS octets) but were otherwise well formed. Switch FW: rx_oversize_pkts TBD - check bug in Shomron	RO
3Ch	31:0	ether_stats_oversize_pkts_low		RO
40h	31:0	ether_stats_fragments_high	The total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). [Internal] counts packet smaller than 64 bytes with FCS error. TBD- mlnx_phy_errors ? Switch FW: rx_stats_fragments	RO
44h	31:0	ether_stats_fragments_low		RO

Table 983 - Ethernet RFC 2819 Counter Group Fields

Offset	Bits	Name	Description	Access
48h	31:0	ether_stats_jabbers_high	The total number of packets received that were longer than MTU octets (excluding framing bits, but including FCS octets), and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). [Internal] counts packet longer than MTU bytes with FCS error. TBD- mlx_phy_errors ? Switch FW: rx_stats_jabbers	RO
4Ch	31:0	ether_stats_jabbers_low		RO
50h	31:0	ether_stats_collisions_high	The best estimate of the total number of collisions on this Ethernet segment. [Internal] Always return zero	RO
54h	31:0	ether_stats_collisions_low		RO
58h	31:0	ether_stats_pkts64octets_high	The total number of packets (including bad packets) received that were 64 octets in length (excluding framing bits but including FCS octets).	RO
5Ch	31:0	ether_stats_pkts64octets_low		RO
60h	31:0	ether_stats_pkts65to127octets_high	The total number of packets (including bad packets) received that were between 65 and 127 octets in length (excluding framing bits but including FCS octets).	RO
64h	31:0	ether_stats_pkts65to127octets_low		RO
68h	31:0	ether_stats_pkts128to255octets_high	The total number of packets (including bad packets) received that were between 128 and 255 octets in length (excluding framing bits but including FCS octets).	RO
6Ch	31:0	ether_stats_pkts128to255octets_low		RO
70h	31:0	ether_stats_pkts256to511octets_high	The total number of packets (including bad packets) received that were between 256 and 511 octets in length (excluding framing bits but including FCS octets).	RO
74h	31:0	ether_stats_pkts256to511octets_low		RO
78h	31:0	ether_stats_pkts512to1023octets_high	The total number of packets (including bad packets) received that were between 512 and 1023 octets in length (excluding framing bits but including FCS octets).	RO
7Ch	31:0	ether_stats_pkts512to1023octets_low		RO
80h	31:0	ether_stats_pkts1024to1518octets_high	The total number of packets (including bad packets) received that were between 1024 and 1518 octets in length (excluding framing bits but including FCS octets).	RO
84h	31:0	ether_stats_pkts1024to1518octets_low		RO
88h	31:0	ether_stats_pkts1519to2047octets_high	The total number of packets (including bad packets) received that were between 1519 and 2047 octets in length (excluding framing bits but including FCS octets).	RO
8Ch	31:0	ether_stats_pkts1519to2047octets_low		RO
90h	31:0	ether_stats_pkts2048to4095octets_high	The total number of packets (including bad packets) received that were between 2048 and 4095 octets in length (excluding framing bits but including FCS octets).	RO
94h	31:0	ether_stats_pkts2048to4095octets_low		RO

Table 983 - Ethernet RFC 2819 Counter Group Fields

Offset	Bits	Name	Description	Access
98h	31:0	ether_stats_pkts4096to8191octets_high	The total number of packets (including bad packets) received that were between 4096 and 8191 octets in length (excluding framing bits but including FCS octets).	RO
9Ch	31:0	ether_stats_pkts4096to8191octets_low		RO
A0h	31:0	ether_stats_pkts8192to10239octets_high	The total number of packets (including bad packets) received that were between 8192 and 10239 octets in length (excluding framing bits but including FCS octets). [Internal] up to MTU (max 10,400 bytes)	RO
A4h	31:0	ether_stats_pkts8192to10239octets_low		RO

28.19.12.4 Ethernet RFC 3635 Counters

Table 984 - Ethernet RFC 3635 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dot3stats_alignment_errors_high																															00h	
dot3stats_alignment_errors_low																															04h	
dot3stats_fcs_errors_high																															08h	
dot3stats_fcs_errors_low																															0Ch	
dot3stats_single_collision_frames_high																															10h	
dot3stats_single_collision_frames_low																															14h	
dot3stats_multiple_collision_frames_high																															18h	
dot3stats_multiple_collision_frames_low																															1Ch	
dot3stats_sqe_test_errors_high																															20h	
dot3stats_sqe_test_errors_low																															24h	
dot3stats_deferred_transmissions_high																															28h	
dot3stats_deferred_transmissions_low																															2Ch	
dot3stats_late_collisions_high																															30h	
dot3stats_late_collisions_low																															34h	
dot3stats_excessive_collisions_high																															38h	
dot3stats_excessive_collisions_low																															3Ch	
dot3stats_internal_mac_transmit_errors_high																															40h	
dot3stats_internal_mac_transmit_errors_low																															44h	
dot3stats_carrier_sense_errors_high																															48h	
dot3stats_carrier_sense_errors_low																															4Ch	

Table 984 - Ethernet RFC 3635 Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dot3stats_frame_too_long_high																																50h
dot3stats_frame_too_long_low																																54h
dot3stats_internal_mac_receive_errors_high																																58h
dot3stats_internal_mac_receive_errors_low																																5Ch
dot3stats_symbol_errors_high																																60h
dot3stats_symbol_errors_low																																64h
dot3control_in_unknown_opcodes_high																																68h
dot3control_in_unknown_opcodes_low																																6Ch
dot3in_pause_frames_high																																70h
dot3in_pause_frames_low																																74h
dot3out_pause_frames_high																																78h
dot3out_pause_frames_low																																7Ch
																																80-F4h

Table 985 - Ethernet RFC 3635 Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	dot3stats_alignment_errors_high	A count of frames received that are not an integral number of octets in length and do not pass the FCS check. [Internal] Always return zero	
04h	31:0	dot3stats_alignment_errors_low		
08h	31:0	dot3stats_fcs_errors_high	A count of frames received that are an integral number of octets in length but do not pass the FCS check. This count does not include frames received with frame-too-long or frame-too-short errors. Switch FW: rerc + received_symbol_errors	
0Ch	31:0	dot3stats_fcs_errors_low		
10h	31:0	dot3stats_single_collision_frames_high	A count of frames that are involved in a single collision, and are subsequently transmitted successfully. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
14h	31:0	dot3stats_single_collision_frames_low		
18h	31:0	dot3stats_multiple_collision_frames_high	A count of frames that are involved in more than one collision and are subsequently transmitted successfully. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
1Ch	31:0	dot3stats_multiple_collision_frames_low		

Table 985 - Ethernet RFC 3635 Counter Group Fields

Offset	Bits	Name	Description	Access
20h	31:0	dot3stats_sqe_test_errors_high	A count of times that the SQE TEST ERROR is received on a particular interface. This counter does not increment on interfaces operating at speeds greater than 10 Mb/s, or on interfaces operating in full-duplex mode. [Internal] Always return zero	
24h	31:0	dot3stats_sqe_test_errors_low		
28h	31:0	dot3stats_deferred_transmissions_high	A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
2Ch	31:0	dot3stats_deferred_transmissions_low		
30h	31:0	dot3stats_late_collisions_high	The number of times that a collision is detected on a particular interface later than one slotTime into the transmission of a packet. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
34h	31:0	dot3stats_late_collisions_low		
38h	31:0	dot3stats_excessive_collisions_high	A count of frames for which transmission on a particular interface fails due to excessive collisions. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
3Ch	31:0	dot3stats_excessive_collisions_low		
40h	31:0	dot3stats_internal_mac_transmit_errors_high	A count of frames for which transmission failed and were discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. [Internal] Always return zero	
44h	31:0	dot3stats_internal_mac_transmit_errors_low		
48h	31:0	dot3stats_carrier_sense_errors_high	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface. This counter does not increment when the interface is operating in full-duplex mode. [Internal] Always return zero	
4Ch	31:0	dot3stats_carrier_sense_errors_low		
50h	31:0	dot3stats_frame_too_long_high	A count of frames received that exceed the maximum permitted frame size. Switch FW: frame_too_long_errors_mtu (includes FCS errors).	
54h	31:0	dot3stats_frame_too_long_low		
58h	31:0	dot3stats_internal_mac_receive_errors_high	A count of frames for which reception failed and were discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. [Internal] Switch FW bug. Currently always return zeros. TBD Should return "buffer_overflow" ?	
5Ch	31:0	dot3stats_internal_mac_receive_errors_low		

Table 985 - Ethernet RFC 3635 Counter Group Fields

Offset	Bits	Name	Description	Access
60h	31:0	dot3stats_symbol_errors_high	The number of times the receiving media is non-idle (a carrier event) for a period of time equal to or greater than minFrameSize, and during which there was at least one occurrence of an event that causes the PHY to indicate 'Receive Error'. [Internal] Switch FW: received_symbol_errors TBD - check with PHY if the counter counts only in a packets.	
64h	31:0	dot3stats_symbol_errors_low		
68h	31:0	dot3control_in_unknown_opcodes_high	A count of MAC Control frames received that contain an opcode that is not supported. [Internal] Switch FW: unsupported_opcodes_received	
6Ch	31:0	dot3control_in_unknown_opcodes_low		
70h	31:0	dot3in_pause_frames_high	A count of MAC Control frames received with an opcode indicating the PAUSE operation. [Internal] Switch FW: frames_received (pause_mac_ctrl_received). TBD - Check if counts PFC	
74h	31:0	dot3in_pause_frames_low		
78h	31:0	dot3out_pause_frames_high	A count of MAC Control frames transmitted with an opcode indicating the PAUSE operation. [Internal] Switch FW: pause_mac_ctrl_transmitted. TBD - Check if counts PFC	
7Ch	31:0	dot3out_pause_frames_low		

28.19.12.5 Ethernet Extended Counters

Table 986 - Ethernet Extended Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
port_transmit_wait_high																															00h	
port_transmit_wait_low																															04h	
ecn_marked_high																															08h	
ecn_marked_low																															0Ch	
no_buffer_discard_mc_high																															10h	
no_buffer_discard_mc_low																															14h	
rx_ebp_high																															18h	
rx_ebp_low																															1Ch	
tx_ebp_high																															20h	
tx_ebp_low																															24h	
rx_buffer_almost_full_high																															28h	

Table 986 - Ethernet Extended Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_buffer_almost_full_low																															2Ch	
rx_buffer_full_high																															30h	
rx_buffer_full_low																															34h	
rx_icrc_encapsulated_high																															38h	
rx_icrc_encapsulated_low																															3Ch	
																															40h-4Ch	
tx_stats_pkts64octets_high																															50h	
tx_stats_pkts64octets_low																															54h	
tx_stats_pkts65to127octets_high																															58h	
tx_stats_pkts65to127octets_low																															5Ch	
tx_stats_pkts128to255octets_high																															60h	
tx_stats_pkts128to255octets_low																															64h	
tx_stats_pkts256to511octets_high																															68h	
tx_stats_pkts256to511octets_low																															6Ch	
tx_stats_pkts512to1023octets_high																															70h	
tx_stats_pkts512to1023octets_low																															74h	
tx_stats_pkts1024to1518octets_high																															78h	
tx_stats_pkts1024to1518octets_low																															7Ch	
tx_stats_pkts1519to2047octets_high																															80h	
tx_stats_pkts1519to2047octets_low																															84h	
tx_stats_pkts2048to4095octets_high																															88h	
tx_stats_pkts2048to4095octets_low																															8Ch	
tx_stats_pkts4096to8191octets_high																															90h	
tx_stats_pkts4096to8191octets_low																															94h	
tx_stats_pkts8192to10239octets_high																															98h	
tx_stats_pkts8192to10239octets_low																															9Ch	
																															A0h-F4h	

Table 987 - Ethernet Extended Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	port_transmit_wait_high	The time (in 4ns resolution) during which the port selected had data to transmit but no data was sent. time = <i>port_transmit_wait</i> *4 [ns] [Internal] Reasons: 1) Flow control 2) PTB (RX) to more than three TQs (HW internal bandwidth). Switch FW : xmit_wait	RO
04h	31:0	port_transmit_wait_low		RO
08h	31:0	ecn_marked_high	A count of packets marked as ECN or potentially marked as ECN. ECN Counting enable/disable is configurable. Valid only for Spectrum family. [Internal] Switch FW : ecn_marking [Internal] See CWGCR.cece	RO
0Ch	31:0	ecn_marked_low		RO
10h	31:0	no_buffer_discard_mc_high	The number of multicast packets dropped due to lack of egress buffer resources. Valid only for Spectrum. [Internal] Switch FW: mc_adm_sb_drop_counters_inc_mc_adm_sb_drop_tq	RO
14h	31:0	no_buffer_discard_mc_low		RO
18h	31:0	rx_ebp_high	The number of received EBP packets.	RO
1Ch	31:0	rx_ebp_low		RO
20h	31:0	tx_ebp_high	The number of transmitted EBP packets.	RO
24h	31:0	tx_ebp_low		RO
28h	31:0	rx_buffer_almost_full_high	The number of events where the port rx buffer has passed a fullness threshold Reserved for Switches.	RO
2Ch	31:0	rx_buffer_almost_full_low		RO
30h	31:0	rx_buffer_full_high	The number of events where the port rx buffer has reached 100% fullness Reserved for Switches.	RO
34h	31:0	rx_buffer_full_low		RO
38h	31:0	rx_icrc_encapsulated_high	The number of roce packets with ICRC error Reserved for Switches.	RO
3Ch	31:0	rx_icrc_encapsulated_low		RO
50h	31:0	tx_stats_pkts64octets_high	The total number of packets (including bad packets) transmitted that were 64 octets in length (excluding framing bits but including FCS octets).	RO
54h	31:0	tx_stats_pkts64octets_low		RO
58h	31:0	tx_stats_pkts65to127octets_high	The total number of packets (including bad packets) transmitted that were between 65 and 127 octets in length (excluding framing bits but including FCS octets).	RO
5Ch	31:0	tx_stats_pkts65to127octets_low		RO

Table 987 - Ethernet Extended Counter Group Fields

Offset	Bits	Name	Description	Access
60h	31:0	tx_stats_pkts128to255octets_high	The total number of packets (including bad packets) transmitted that were between 128 and 255 octets in length (excluding framing bits but including FCS octets).	RO
64h	31:0	tx_stats_pkts128to255octets_low		RO
68h	31:0	tx_stats_pkts256to511octets_high	The total number of packets (including bad packets) transmitted that were between 256 and 511 octets in length (excluding framing bits but including FCS octets).	RO
6Ch	31:0	tx_stats_pkts256to511octets_low		RO
70h	31:0	tx_stats_pkts512to1023octets_high	The total number of packets (including bad packets) transmitted that were between 512 and 1023 octets in length (excluding framing bits but including FCS octets).	RO
74h	31:0	tx_stats_pkts512to1023octets_low		RO
78h	31:0	tx_stats_pkts1024to1518octets_high	The total number of packets (including bad packets) transmitted that were between 1024 and 1518 octets in length (excluding framing bits but including FCS octets).	RO
7Ch	31:0	tx_stats_pkts1024to1518octets_low		RO
80h	31:0	tx_stats_pkts1519to2047octets_high	The total number of packets (including bad packets) transmitted that were between 1519 and 2047 octets in length (excluding framing bits but including FCS octets).	RO
84h	31:0	tx_stats_pkts1519to2047octets_low		RO
88h	31:0	tx_stats_pkts2048to4095octets_high	The total number of packets (including bad packets) transmitted that were between 2048 and 4095 octets in length (excluding framing bits but including FCS octets).	RO
8Ch	31:0	tx_stats_pkts2048to4095octets_low		RO
90h	31:0	tx_stats_pkts4096to8191octets_high	The total number of packets (including bad packets) transmitted that were between 4096 and 8191 octets in length (excluding framing bits but including FCS octets).	RO
94h	31:0	tx_stats_pkts4096to8191octets_low		RO
98h	31:0	tx_stats_pkts8192to10239octets_high	The total number of packets (including bad packets) transmitted that were between 8192 and 10239 octets in length (excluding framing bits but including FCS octets).transmitted [Internal] up to MTU (max 10,400 bytes)	RO
9Ch	31:0	tx_stats_pkts8192to10239octets_low		RO

28.19.12.6 Ethernet Discard Counters.

Table 988 - Ethernet Discard Counters Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ingress_general_high																															00h	

Table 988 - Ethernet Discard Counters Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ingress_discard_all_low																															7Ch	
																																80h

Table 989 - Ethernet Discard Counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	ingress_general_high	Ingress general [Internal] In switch counts the following: 1) switch decap encap prevention 2) ingress spanning tree 3) smac is mc 4) smac equal dmac 5) ieee reserved mac provider bridge group (dmac == 01-80-C2-00-00-08) 6) ethertype invalid zone 7) l2 type none (after decap – no L2) 8) empty tq list error (for MC packets, e.g. due to port isolation and ports down) 9) decap of more than 126 bytes 10) decap remains with short packet 11) vxlan port isolation 12) mc count wraparound (internal HW error) 13) tx base version (from SW) 14) tx base type (from SW) 15) tx base system port mid (from SW)	RO
04h	31:0	ingress_general_low		
08h	31:0	ingress_policy_engine_high	Ingress policy engine discards	RO
0Ch	31:0	ingress_policy_engine_low		
10h	31:0	ingress_vlan_membership_high	Ingress VLAN membership filter discards	RO
14h	31:0	ingress_vlan_membership_low		
18h	31:0	ingress_tag_frame_type_high	Ingress VLAN tag allowance (tagged, untagged and prio-tagged) filter discards	RO
1Ch	31:0	ingress_tag_frame_type_low		
20h	31:0	egress_vlan_membership_high	Egress VLAN membership filter discards	RO
24h	31:0	egress_vlan_membership_low		

Table 989 - Ethernet Discard Counter Group Fields

Offset	Bits	Name	Description	Access
28h	31:0	loopback_filter_high	Loop-back filter discards	RO
2Ch	31:0	loopback_filter_low		
30h	31:0	egress_general_high	Egress general discards [Internal] in Spectrum counts only MTU discards	RO
34h	31:0	egress_general_low		
38h	31:0	egress_link_down_high	Egress link down discards [Internal] Currently doesn't counts anything , return always 0. TBD - should be internal or removed from PRM (SDK)	RO
3Ch	31:0	egress_link_down_low		
40h	31:0	egress_hoq_high	Head-of-Queue time-out discards	RO
44h	31:0	egress_hoq_low		
48h	31:0	port_isolation_high	Port isolation filter discards [Internal] Not supported due to HW bug.	RO
4Ch	31:0	port_isolation_low		
50h	31:0	egress_policy_engine_high	Egress policy engine discards	RO
54h	31:0	egress_policy_engine_low		
58h	31:0	ingress_tx-link_down_high	Per ingress port count dropped packets due to egress link down	RO
5Ch	31:0	ingress_tx_link_down_low		
60h	31:0	egress_stp_filter_high	Egress spanning tree filter	RO
64h	31:0	egress_stp_filter_low		
68h	31:0	egress_hoq_stall_high	Number of sequential packets dropped, due to Head-Of-Queue Lifetime Limit, that causes the port to enter the Stall state Reserved for Switches	RO
6Ch	31:0	egress_hoq_stall_low		
70h	31:0	egress_sll_high	Number of packets dropped, because the Switch Lifetime Limit was exceeded. Applies for switches only. Reserved for HCAs	RO
74h	31:0	egress_sll_low		
78h	31:0	ingress_discard_all_high	Number of packets dropped, because of any reason. Note: see description in the description area for limitations of this counter. Applies for switches only. Reserved for HCAs	RO
7Ch	31:0	ingress_discard_all_low		

28.19.12.7 Ethernet Per Priority Counters

NOTE: Ingress counters are supported only up to prio 7 (8 priors)

Table 990 - Ethernet Per Priority Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_octets_high																																00h
rx_octets_low																																04h
rx_uc_frames_high																																08h
rx_uc_frames_low																																0Ch
rx_mc_frames_high																																10h
rx_mc_frames_low																																14h
rx_bc_frames_high																																18h
rx_bc_frames_low																																1Ch
rx_frames_high																																20h
rx_frames_low																																24h
tx_octets_high																																28h
tx_octets_low																																2Ch
tx_uc_frames_high																																30h
tx_uc_frames_low																																34h
tx_mc_frames_high																																38h
tx_mc_frames_low																																3Ch
tx_bc_frames_high																																40h
tx_bc_frames_low																																44h
tx_frames_high																																48h
tx_frames_low																																4Ch
rx_pause_high																																50h
rx_pause_low																																54h
rx_pause_duration_high																																58h
rx_pause_duration_low																																5Ch
tx_pause_high																																60h
tx_pause_low																																64h
tx_pause_duration_high																																68h
tx_pause_duration_low																																6Ch
rx_pause_transition_high																																70h

Table 990 - Ethernet Per Priority Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_pause_transition_low																																74h
rx_discards_high																																78h
rx_discards_low																																7Ch
device_stall_minor_watermark_cnt_high																																80h
device_stall_minor_watermark_cnt_low																																84h
device_stall_critical_watermark_cnt_high																																88h
device_stall_critical_watermark_cnt_low																																8Ch
																																90h-F4h

Table 991 - Ethernet Per Priority Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	rx_octets_high	The total number of octets received, including framing characters.	RO
04h	31:0	rx_octets_low	[Internal] Switch FW: in_octets_prio<x>. Includes discard due to buffer overflow, control (pause) frames. Counts also Ethernet header and FCS. Doesn't count bad packets	RO
08h	31:0	rx_uc_frames_high	The number of packets delivered which were not addressed to a multicast or broadcast address.	RO
0Ch	31:0	rx_uc_frames_low		RO
10h	31:0	rx_mc_frames_high	The number of packets delivered which were addressed to a multicast address.	RO
14h	31:0	rx_mc_frames_low		RO
18h	31:0	rx_bc_frames_high	The number of packets delivered which were addressed to a broadcast address.	RO
1Ch	31:0	rx_bc_frames_low		RO
20h	31:0	rx_frames_high	The total number of packets received for this priority (including control frames). [Internal] Switch FW: in_pkts_prio<x>. Includes discard due to buffer overflow, control (pause) frames. Doesn't count bad packets [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
24h	31:0	rx_frames_low		RO
28h	31:0	tx_octets_high	The total number of octets transmitted, including framing characters. [Internal] Includes control (pause) frames - TBD TBD - check with Kriss regarding control (pause) frames. FW: port_xmit_data<x>	RO
2Ch	31:0	tx_octets_low		RO

Table 991 - Ethernet Per Priority Group Fields

Offset	Bits	Name	Description	Access
30h	31:0	tx_uc_frames_high	The total number of packets transmitted which were not addressed to a multicast or broadcast address.	RO
34h	31:0	tx_uc_frames_low		RO
38h	31:0	tx_mc_frames_high	The total number of packets transmitted which were addressed to a multi-cast address.	RO
3Ch	31:0	tx_mc_frames_low		RO
40h	31:0	tx_bc_frames_high	The total number of packets transmitted which were addressed to a broad-cast address.	RO
44h	31:0	tx_bc_frames_low		RO
48h	31:0	tx_frames_high	The total number of packets transmitted. [Internal] Should Includes control (pause) frames - TBD Switch FW: port_xmit_pkts_prio<x> [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
4Ch	31:0	tx_frames_low		RO
50h	31:0	rx_pause_high	The total number of PAUSE frames received from the far-end port. Switch FW: rpause<x> [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
54h	31:0	rx_pause_low		RO
58h	31:0	rx_pause_duration_high	The total time in microseconds that transmission of packets to the far-end port have been paused. Switch FW: rx_pause_dura_prio<x> TBD - Possible FW bug regarding the us units. [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
5Ch	31:0	rx_pause_duration_low		RO
60h	31:0	tx_pause_high	The total number of PAUSE(counted on prio0) or PFC frames sent to the far-end port. Switch FW: tpause<x> TBD - check if Global is counted on prio0 [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
64h	31:0	tx_pause_low		RO
68h	31:0	tx_pause_duration_high	The total time in microseconds that the far-end port have been requested to pause. Switch FW: tx_pause_dura_prio<x> TBD - Possible FW bug regarding the us units. [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
6Ch	31:0	tx_pause_duration_low		RO
70h	31:0	rx_pause_transi- tion_high	Counts the number of transitions from Xoff to Xon. [Internal] Switch FW: tx_pause_transition_prio<x> [Internal] Spectrum: HW bug, always 0 [Internal] NIC : return tx instead of rx - possible bug also.	RO
74h	31:0	rx_pause_transition_low		RO

Table 991 - Ethernet Per Priority Group Fields

Offset	Bits	Name	Description	Access
78h	31:0	rx_discards_high	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. <i>[Internal] TBD add a note of which devices support this counter.</i> FW: in_discards_prio<i>. Approved CCB march 2016. HW : 1. Headroom full: mtu fragment prevention check For lossless: hearoom is too small (by RTT) For lossy: headroom is too small (< 2MTU) or pipeline latency 2. discard due to force overflow configuration - FW closes ingress	RO
7Ch	31:0	rx_discards_low		RO
80h	31:0	device_stall_minor_watermark_cnt_high	The number of times the device detected a stalled state for a period longer than <i>device_stall_minor_watermark</i> The counter is presented in priority 0, but is a sum of all events on all priorities (including global pause).	RO
84h	31:0	device_stall_minor_watermark_cnt_low		RO
88h	31:0	device_stall_critical_watermark_cnt_high	The number of times the device detected a stalled state for a period longer than <i>device_stall_critical_watermark</i> The counter is presented in priority 0, but is a sum of all events on all priorities (including global pause).	RO
8Ch	31:0	device_stall_critical_watermark_cnt_low		RO

28.19.12.8 [Internal] Ethernet Out Of Band Port Counters Data

Table 992 - Ethernet OOB Port Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
if_in_ucast_pkts_high																															00h	
if_in_ucast_pkts_low																															04h	
if_in_octets_high																															08h	
if_in_octets_low																															0Ch	
if_in_discards_high																															10h	
if_in_discards_low																															14h	
if_in_errors_high																															18h	
if_in_errors_low																															1Ch	
ether_stats_crc_align_errors_high																															20h	
ether_stats_crc_align_errors_low																															24h	
ether_stats_undersize_pkts_high																															28h	
ether_stats_undersize_pkts_low																															2Ch	
ether_stats_oversize_pkts_high																															30h	
ether_stats_oversize_pkts_low																															34h	

Table 993 - Ethernet OOB Port Counters Data Fields

Offset	Bits	Name	Description	Access
08h	31:0	if_in_octets_high	The total number of octets received, including framing characters. Including MAC control frames. [Internal] Includes discard due to buffer overflow. Includes control (pause) frames. Counts also Ethernet header and FCS. FW: octets_received_ok_global	RO
0Ch	31:0	if_in_octets_low		RO
10h	31:0	if_in_discards_high	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. FW: in_discards_global HW : TBD	RO
14h	31:0	if_in_discards_low		RO
18h	31:0	if_in_errors_high	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. (FCS, Phy, over/user size and length). Switch FW: received_symbols_errors+rcrc+rx_stats_undersize_pkts+frames_too_long_errors_mtu+in_range_length_error+out_of_range_length	RO
1Ch	31:0	if_in_errors_low		RO
20h	31:0	ether_stats_crc_align_errors_high	The total number of packets received that had a length (excluding framing bits, but including FCS octets) of between 64 and MTU octets, inclusive, but had either a bad frame check sequence (FCS) with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Switch FW: rcrc + received_symbol_errors	RO
24h	31:0	ether_stats_crc_align_errors_low		RO
28h	31:0	ether_stats_undersize_pkts_high	The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. Switch FW: rx_stats_undersize_pkts	RO
2Ch	31:0	ether_stats_undersize_pkts_low		RO
30h	31:0	ether_stats_oversize_pkts_high	The total number of packets received that were longer than MTU octets (excluding framing bits, but including FCS octets) but were otherwise well formed. Switch FW: rx_oversize_pkts	RO
34h	31:0	ether_stats_oversize_pkts_low		RO
38h	31:0	dot3stats_symbol_errors_high	The number of times the receiving media is non-idle (a carrier event) for a period of time equal to or greater than minFrameSize, and during which there was at least one occurrence of an event that causes the PHY to indicate 'Receive Error'. [Internal] Switch FW: received_symbol_errors	RO
3Ch	31:0	dot3stats_symbol_errors_low		RO
40h	31:0	dot3in_pause_frames_high	A count of MAC Control frames received with an opcode indicating the PAUSE operation. [Internal] Switch FW: frames_received (pause_mac_ctrl_received). TBD - Check if counts PFC	RO
44h	31:0	dot3in_pause_frames_low		RO

Table 993 - Ethernet OOB Port Counters Data Fields

Offset	Bits	Name	Description	Access
48h	31:0	dot3control_in_unknown_opcodes_high	A count of MAC Control frames received that contain an opcode that is not supported. [Internal] Switch FW: unsupported_opcodes_received	RO
4Ch	31:0	dot3control_in_unknown_opcodes_low		RO
50h	31:0	if_out_ucast_pkts_high	The total number of packets that higher-level protocols requested be transmitted and were not addressed to a multicast or broadcast MAC address, including those that were discarded or not sent. [Internal] Not including Egress policer Switch FW: port_unicast_xmit_pkts	RO
54h	31:0	if_out_ucast_pkts_low		RO
58h	31:0	if_out_octets_high	The total number of octets transmitted out of the interface, including framing characters. [Internal] Includes control (pause) frames. FW: port_xmit_data + 64*tpause_global	RO
5Ch	31:0	if_out_octets_low		RO
60h	31:0	dot3out_pause_frames_high	A count of MAC Control frames transmitted with an opcode indicating the PAUSE operation. [Internal] Switch FW: pause_mac_ctrl_transmitted. TBD - Check if counts PFC	RO
64h	31:0	dot3out_pause_frames_low		RO
68h	31:0	if_out_errors_high	The number of outbound packets that could not be transmitted because of errors. Switch FW: discard_mtu	RO
6Ch	31:0	if_out_errors_low		RO
70h	31:0	phy_time_since_last_clear_high	The time passed since the last phy counters clear event in msec.	RO
74h	31:0	phy_time_since_last_clear_low		RO
78h	31:0	phy_received_bits_high	This counter provides information on the total amount of traffic (bits) received.	RO
7Ch	31:0	phy_received_bits_low		RO
80h	31:0	phy_symbol_errors_high	This counter provides information on error bits.	RO
84h	31:0	phy_symbol_errors_low		RO

28.19.12.9 Ethernet Per Traffic Class Counters

Table 994 - Ethernet Per Traffic Class Group data layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
transmit_queue_high																															00h	
transmit_queue_low																															04h	
no_buffer_discard_uc_high																															08h	
no_buffer_discard_uc_low																															0Ch	

Table 994 - Ethernet Per Traffic Class Group data layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																10h-F4h

Table 995 - Ethernet Per Traffic Class Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	transmit_queue_high	Contains the transmit queue depth in bytes on traffic class selected by <i>traffic_class</i> of the port selected by <i>local_port</i> . [Internal] The goal of measuring queue depths is to enable software to compute the average time data waits for transmission inside a node. This is a gauge (can't be cleared). TBD - should be presented in Cells or Bytes. Open on Zvi should define until 24/07/16. Zachy wants Cells, Need to check with Ahmad. Need to check if this gauge is relevant to NIC. FW currently working in Cells [Internal]: This field is valid for OOB port if <i>pnat</i> field is set to '3 - OOB port number'.	RO
04h	31:0	transmit_queue_low		RO
08h	31:0	no_buffer_discard_uc_high	The number of unicast packets dropped due to lack of shared buffer resources. Valid only for Spectrum. Valid for HCA when <i>ppcnt_no_buffer_uc</i> is set in PCAM.	RO
0Ch	31:0	no_buffer_discard_uc_low		RO

28.19.12.10 Ethernet Per Traffic Class Congestion Counters

Table 996 - Ethernet Per Traffic Class Congestion Group data layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
wred_discard_high																															00h	
wred_discard_low																															04h	
ecn_marked_tc_high																															08h	
ecn_marked_tc_low																															0Ch	
																																10h-F4h

Table 997 - Ethernet Per Traffic Class Congestion Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	wred_discard_high	The number of packet that are dropped by the Weighted Random Early Detection (WRED) function. For HCAs - valid when PCAM.feature_cap_mask "ppent eth per tc cong group supported" is set.	RO
04h	31:0	wred_discard_low		RO
08h	31:0	ecn_marked_tc_high	A count of packets marked as ECN or potentially marked as ECN. ECN Counting enable/disable is configurable. For HCAs - valid when PCAM.feature_cap_mask "ppent eth per tc cong group supported" is set. [Internal] Switch FW: ecn_marking [Internal] See CWGCR.cece	RO
0Ch	31:0	ecn_marked_tc_low		RO

28.19.12.11 [Internal] Ethernet Per Receive Buffer Counters

Table 998 - Ethernet Per Receive Buffer Group data layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_no_buffer_discard_uc_high																																00h
rx_no_buffer_discard_uc_low																																04h
																																08h-F4h

Table 999 - Ethernet Per Receive Buffer Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	rx_no_buffer_discard_uc_high	The number of unicast packets dropped due to lack of ingress shared buffer resources. Not supported by Spectrum. Supported by Spectrum-2.	RO
04h	31:0	rx_no_buffer_discard_uc_low		RO

28.19.12.12 Link Level Retransmission Counters

Table 1000 - Link Level Retransmission Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
llr_rx_cells_high																																00h
llr_rx_cells_low																																04h
llr_rx_error_high																																08h
llr_rx_error_low																																0Ch

Table 1000 - Link Level Retransmission Counter Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
llr_rx_crc_error_high																															10h	
llr_rx_crc_error_low																															14h	
llr_tx_cells_high																															18h	
llr_tx_cells_low																															1Ch	
llr_tx_ret_cells_high																															20h	
llr_tx_ret_cells_low																															24h	
llr_tx_ret_events_high																															28h	
llr_tx_ret_events_low																															2Ch	
																																30-F4h

Table 1001 - Link Level Retransmission Counter Group Data Fields

Offset	Bits	Name	Description	Access
00h	31:0	llr_rx_cells_high	The total number of cells received (including bad cells).	RO
04h	31:0	llr_rx_cells_low		
08h	31:0	llr_rx_error_high	The total number of bad cells received (including CRC error). [Internal] This counter includes all tail errors	RO
0Ch	31:0	llr_rx_error_low		
10h	31:0	llr_rx_crc_error_high	The total number of cells received with CRC error.	RO
14h	31:0	llr_rx_crc_error_low		
18h	31:0	llr_tx_cells_high	The total number of cells transmitted (including retransmitted cells).	RO
1Ch	31:0	llr_tx_cells_low		
20h	31:0	llr_tx_ret_cells_high	The total number of cells retransmitted.	RO
24h	31:0	llr_tx_ret_cells_low		
28h	31:0	llr_tx_ret_events_high	The total number of retransmission events.	RO
2Ch	31:0	llr_tx_ret_events_low		

28.19.12.13 InfiniBand PortCounters Attribute of the Performance Class Counters

Table 1002 - InfiniBand PortCounters Attribute Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
symbol_error_counter																link_error_recovery_counter										link_downed_counter										00h

Table 1003 - InfiniBand PortCounters Attribute Group Fields

Offset	Bits	Name	Description	Access
00h	31:16	symbol_error_counter	For counter description please refer to the InfiniBand specification	RO
	15:8	link_error_recovery_counter		RO
	7:0	link_downed_counter		RO
04h	31:16	port_rcv_errors		RO
	15:0	port_rcv_remote_physical_errors		RO
08h	31:16	port_rcv_switch_relay_errors		RO
	15:0	port_xmit_discards		RO
0Ch	31:24	port_xmit_constraint_errors		RO
	23:16	port_rcv_constraint_errors		RO
	7:4	local_link_integrity_errors		RO
	3:0	excessive_buffer_overflow_errors		RO
10h	15:0	vl_15_dropped		RO
14h	31:0	port_xmit_data		RO
18h	31:0	port_rcv_data		RO
1Ch	31:0	port_xmit_pkts		RO
20h	31:0	port_rcv_pkts		RO
24h	31:0	port_xmit_wait		RO

28.19.12.14 InfiniBand Extended PortCounters Attribute of the Performance Class Counters

Table 1004 - InfiniBand Extended PortCounters Attribute Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
symbol_error_counter_high																															00h	
symbol_error_counter_low																															04h	
link_error_recovery_counter_high																															08h	
link_error_recovery_counter_low																															0Ch	

Table 1004 - InfiniBand Extended PortCounters Attribute Group Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
port_effective_rcv_pkts_high																															88h	
port_effective_rcv_pkts_low																															8Ch	
port_effective_rcv_data_high																															90h	
port_effective_rcv_data_low																															94h	
																																98h-F4h

Table 1005 - InfiniBand Extended PortCounters Attribute Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	symbol_error_counter_high	For counter description please refer to the InfiniBand specification	RO
04h	31:0	symbol_error_counter_low		RO
08h	31:0	link_error_recovery_counter_high	For counter description please refer to the InfiniBand specification	RO
0Ch	31:0	link_error_recovery_counter_low		RO
10h	31:0	link_downed_counter_high	For counter description please refer to the InfiniBand specification	RO
14h	31:0	link_downed_counter_low		RO
18h	31:0	port_rcv_errors_high	For counter description please refer to the InfiniBand specification	RO
1Ch	31:0	port_rcv_errors_low		RO
20h	31:0	port_rcv_remote_physical_errors_high	For counter description please refer to the InfiniBand specification	RO
24h	31:0	port_rcv_remote_physical_errors_low		RO
28h	31:0	port_rcv_switch_relay_errors_high	For counter description please refer to the InfiniBand specification	RO
2Ch	31:0	port_rcv_switch_relay_errors_low		RO
30h	31:0	port_xmit_discards_high	For counter description please refer to the InfiniBand specification	RO
34h	31:0	port_xmit_discards_low		RO

Table 1005 - InfiniBand Extended PortCounters Attribute Group Fields

Offset	Bits	Name	Description	Access
38h	31:0	port_xmit_constraint_errors_high	For counter description please refer to the InfiniBand specification	RO
3Ch	31:0	port_xmit_constraint_errors_low		RO
40h	31:0	port_rcv_constraint_errors_high	For counter description please refer to the InfiniBand specification	RO
44h	31:0	port_rcv_constraint_errors_low		RO
48h	31:0	local_link_integrity_errors_high	For counter description please refer to the InfiniBand specification	RO
4Ch	31:0	local_link_integrity_errors_low		RO
50h	31:0	excessive_buffer_overflow_errors_high	For counter description please refer to the InfiniBand specification	RO
54h	31:0	excessive_buffer_overflow_errors_low		RO
58h	31:0	vl_15_dropped_high	For counter description please refer to the InfiniBand specification	RO
5Ch	31:0	vl_15_dropped_low		RO
60h	31:0	port_xmit_data_high	For counter description please refer to the InfiniBand specification	RO
64h	31:0	port_xmit_data_low		RO
68h	31:0	port_rcv_data_high	For counter description please refer to the InfiniBand specification	RO
6Ch	31:0	port_rcv_data_low		RO
70h	31:0	port_xmit_pkts_high	For counter description please refer to the InfiniBand specification	RO
74h	31:0	port_xmit_pkts_low		RO
78h	31:0	port_rcv_pkts_high	For counter description please refer to the InfiniBand specification	RO
7Ch	31:0	port_rcv_pkts_low		RO
80h	31:0	port_xmit_wait_high	For counter description please refer to the InfiniBand specification	RO
84h	31:0	port_xmit_wait_low		RO
88h	31:0	port_effective_rcv_pkts_high		RO
8Ch	31:0	port_effective_rcv_pkts_low		RO

Table 1006 - Physical Layer Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																C8-F4h

Table 1007 - Physical Layer Counters Fields

Offset	Bits	Name	Description	Access
00h	31:0	time_since_last_clear_high	The time passed since the last counters clear event in msec. [Internal] supported in USR	RO
04h	31:0	time_since_last_clear_low		
08h	31:0	symbol_errors_high	Perf.PortCounters(SymbolErrorCounter) = Perf.Port-Counters(UnknownSymbol) [Internal] supported in USR	RO
0Ch	31:0	symbol_errors_low		
10h	31:0	sync_headers_errors_high	Perf.PortExtendedSpeedsCounters(SynchHeader-ErrorCounter)	RO
14h	31:0	sync_headers_errors_low		
18h	31:0	edpl_bip_errors_lane0_high	Perf.PortExtendedSpeedsCounters(ErrorDetectionPer-Lane0Counter) / BIP error counter, lane 0 In PRBS test mode, indicates the number of PRBS errors on lane 0 [Internal] supported in USR	RO
1Ch	31:0	edpl_bip_errors_lane0_low		
20h	31:0	edpl_bip_errors_lane1_high	Perf.PortExtendedSpeedsCounters(ErrorDetectionPer-Lane1Counter) / BIP error counter, lane 1 In PRBS test mode, indicates the number of PRBS errors on lane 1 [Internal] supported in USR	RO
24h	31:0	edpl_bip_errors_lane1_low		
28h	31:0	edpl_bip_errors_lane2_high	Perf.PortExtendedSpeedsCounters(ErrorDetectionPer-Lane2Counter) / BIP error counter, lane 2 In PRBS test mode, indicates the number of PRBS errors on lane 2 [Internal] supported in USR	RO
2Ch	31:0	edpl_bip_errors_lane2_low		
30h	31:0	edpl_bip_errors_lane3_high	Perf.PortExtendedSpeedsCounters(ErrorDetectionPer-Lane3Counter) / BIP error counter, lane 3 In PRBS test mode, indicates the number of PRBS errors on lane 3 [Internal] supported in USR	RO
34h	31:0	edpl_bip_errors_lane3_low		
38h	31:0	fc_fec_corrected_blocks_lane0_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane0) [Internal] supported in USR - ZL FEC	RO
3Ch	31:0	fc_fec_corrected_blocks_lane0_low		
40h	31:0	fc_fec_corrected_blocks_lane1_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane1) [Internal] supported in USR - ZL FEC	RO
44h	31:0	fc_fec_corrected_blocks_lane1_low		

Table 1007 - Physical Layer Counters Fields

Offset	Bits	Name	Description	Access
48h	31:0	fc_fec_corrected_blocks_lane2_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane2) [Internal] supported in USR - ZL FEC	RO
4Ch	31:0	fc_fec_corrected_blocks_lane2_low		
50h	31:0	fc_fec_corrected_blocks_lane3_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane3) [Internal] supported in USR - ZL FEC	RO
54h	31:0	fc_fec_corrected_blocks_lane3_low		
58h	31:0	fc_fec_uncorrectable_blocks_lane0_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane0) [Internal] supported in USR - ZL FEC	RO
5Ch	31:0	fc_fec_uncorrectable_blocks_lane0_low		
60h	31:0	fc_fec_uncorrectable_blocks_lane1_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane1) [Internal] supported in USR - ZL FEC	RO
64h	31:0	fc_fec_uncorrectable_blocks_lane1_low		
68h	31:0	fc_fec_uncorrectable_blocks_lane2_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane2) [Internal] supported in USR - ZL FEC	RO
6Ch	31:0	fc_fec_uncorrectable_blocks_lane2_low		
70h	31:0	fc_fec_uncorrectable_blocks_lane3_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane3) [Internal] supported in USR - ZL FEC	RO
74h	31:0	fc_fec_uncorrectable_blocks_lane3_low		
78h	31:0	rs_fec_corrected_blocks_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounter)	RO
7Ch	31:0	rs_fec_corrected_blocks_low		
80h	31:0	rs_fec_uncorrectable_blocks_high	Perf.PortExtendedSpeedsCounters(FEUncorrectableBlockCounter)	RO
84h	31:0	rs_fec_uncorrectable_blocks_low		
88h	31:0	rs_fec_no_errors_blocks_high	The number of RS-FEC blocks received that had no errors. Note: Total rs_fec blocks received = rs_fec_corrected_blocks + rs_fec_uncorrectable_blocks + rs_fec_no_errors_blocks	RO
8Ch	31:0	rs_fec_no_errors_blocks_low		
90h	31:0	rs_fec_single_error_blocks_high	The number of corrected RS-FEC blocks received that had exactly 1 error symbol. Note: A FEC block that is counted by rs_fec_single_error_blocks is counted also by rs_fec_corrected_blocks	RO
94h	31:0	rs_fec_single_error_blocks_low		
98h	31:0	rs_fec_corrected_symbols_total_high	Perf.PortExtendedSpeedCounters(PortFECCorrectedSymbolCounter)	RO
9Ch	31:0	rs_fec_corrected_symbols_total_low		
A0h	31:0	rs_fec_corrected_symbols_lane0_high	Perf.PortExtendedSpeedCounters(FECCorrectedSymbolCounterLane0)	RO
A4h	31:0	rs_fec_corrected_symbols_lane0_low		
A8h	31:0	rs_fec_corrected_symbols_lane1_high	Perf.PortExtendedSpeedCounters(FECCorrectedSymbolCounterLane1)	RO
ACh	31:0	rs_fec_corrected_symbols_lane1_low		

Table 1007 - Physical Layer Counters Fields

Offset	Bits	Name	Description	Access
B0h	31:0	rs_fec_corrected_symbols_lane2_high	Perf.PortExtendedSpeedCounters(FECCorrectedSymbolCounterLane2)	RO
B4h	31:0	rs_fec_corrected_symbols_lane2_low		
B8h	31:0	rs_fec_corrected_symbols_lane3_high	Perf.PortExtendedSpeedCounters(FECCorrectedSymbolCounterLane3)	RO
BCh	31:0	rs_fec_corrected_symbols_lane3_low		
C0h	31:0	link_down_events	Perf.PortCounters(LinkDownedCounter) [Internal] supported in USR - ZL FEC	RO
C4h	31:0	successful_recovery_events	Perf.PortCounters(LinkErrorRecoveryCounter) Note: This counter is valid only for IB.	RO

28.19.12.16 USR_XSR Physical Layer Counters Group

Table 1008 - USR_XSR Physical Layer Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
time_since_last_clear_high																																00h
time_since_last_clear_low																																04h
fc_fec_corrected_blocks_lane0_high																																08h
fc_fec_corrected_blocks_lane0_low																																0Ch
fc_fec_corrected_blocks_lane1_high																																10h
fc_fec_corrected_blocks_lane1_low																																14h
fc_fec_corrected_blocks_lane2_high																																18h
fc_fec_corrected_blocks_lane2_low																																1Ch
fc_fec_corrected_blocks_lane3_high																																20h
fc_fec_corrected_blocks_lane3_low																																24h
fc_fec_corrected_blocks_lane4_high																																28h
fc_fec_corrected_blocks_lane4_low																																2Ch
fc_fec_corrected_blocks_lane5_high																																30h
fc_fec_corrected_blocks_lane5_low																																34h
fc_fec_corrected_blocks_lane6_high																																38h
fc_fec_corrected_blocks_lane6_low																																3Ch
fc_fec_corrected_blocks_lane7_high																																40h
fc_fec_corrected_blocks_lane7_low																																44h
fc_fec_uncorrectable_blocks_lane0_high																																48h

Table 1009 - USR_XSR Physical Layer Counters Fields

Offset	Bits	Name	Description	Access
20h	31:0	fc_fec_corrected_blocks_lane3_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane3) [Internal] supported in USR - ZL FEC	RO
24h	31:0	fc_fec_corrected_blocks_lane3_low		
28h	31:0	fc_fec_corrected_blocks_lane4_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane4) [Internal] supported in USR - ZL FEC	RO
2Ch	31:0	fc_fec_corrected_blocks_lane4_low		
30h	31:0	fc_fec_corrected_blocks_lane5_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane5) [Internal] supported in USR - ZL FEC	RO
34h	31:0	fc_fec_corrected_blocks_lane5_low		
38h	31:0	fc_fec_corrected_blocks_lane6_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane6) [Internal] supported in USR - ZL FEC	RO
3Ch	31:0	fc_fec_corrected_blocks_lane6_low		
40h	31:0	fc_fec_corrected_blocks_lane7_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane7) [Internal] supported in USR - ZL FEC	RO
44h	31:0	fc_fec_corrected_blocks_lane7_low		
48h	31:0	fc_fec_uncorrectable_blocks_lane0_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane0) [Internal] supported in USR - ZL FEC	RO
4Ch	31:0	fc_fec_uncorrectable_blocks_lane0_low		
50h	31:0	fc_fec_uncorrectable_blocks_lane1_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane1) [Internal] supported in USR - ZL FEC	RO
54h	31:0	fc_fec_uncorrectable_blocks_lane1_low		
58h	31:0	fc_fec_uncorrectable_blocks_lane2_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane2) [Internal] supported in USR - ZL FEC	RO
5Ch	31:0	fc_fec_uncorrectable_blocks_lane2_low		
60h	31:0	fc_fec_uncorrectable_blocks_lane3_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane3) [Internal] supported in USR - ZL FEC	RO
64h	31:0	fc_fec_uncorrectable_blocks_lane3_low		
68h	31:0	fc_fec_uncorrectable_blocks_lane4_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane4) [Internal] supported in USR - ZL FEC	RO
6Ch	31:0	fc_fec_uncorrectable_blocks_lane4_low		
70h	31:0	fc_fec_uncorrectable_blocks_lane5_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane5) [Internal] supported in USR - ZL FEC	RO
74h	31:0	fc_fec_uncorrectable_blocks_lane5_low		
78h	31:0	fc_fec_uncorrectable_blocks_lane6_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane6) [Internal] supported in USR - ZL FEC	RO
7Ch	31:0	fc_fec_uncorrectable_blocks_lane6_low		
80h	31:0	fc_fec_uncorrectable_blocks_lane7_high	Perf.PortExtendedSpeedsCounters(FECCorrectableBlockCounterLane7) [Internal] supported in USR - ZL FEC	RO
84h	31:0	fc_fec_uncorrectable_blocks_lane7_low		
88h	31:0	link_down_events	Perf.PortCounters(LinkDownedCounter) [Internal] supported in USR	RO

28.19.12.17Physical Layer statistical Counters Group

Table 1010 - Physical Layer Statistical Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
time_since_last_clear_high																															00h	
time_since_last_clear_low																															04h	
phy_received_bits_high																															08h	
phy_received_bits_low																															0Ch	
phy_symbol_errors_high																															10h	
phy_symbol_errors_low																															14h	
phy_corrected_bits_high																															18h	
phy_corrected_bits_low																															1Ch	
phy_raw_errors_lane0_high																															20h	
phy_raw_errors_lane0_low																															24h	
phy_raw_errors_lane1_high																															28h	
phy_raw_errors_lane1_low																															2Ch	
phy_raw_errors_lane2_high																															30h	
phy_raw_errors_lane2_low																															34h	
phy_raw_errors_lane3_high																															38h	
phy_raw_errors_lane3_low																															3Ch	
phy_raw_errors_lane4_high																															40h	
phy_raw_errors_lane4_low																															44h	
phy_raw_errors_lane5_high																															48h	
phy_raw_errors_lane5_low																															4Ch	
phy_raw_errors_lane6_high																															50h	
phy_raw_errors_lane6_low																															54h	
phy_raw_errors_lane7_high																															58h	
phy_raw_errors_lane7_low																															5Ch	
																raw_ber_magnitude									raw_ber_coef					60h		
																effective_ber_magnitude									effec- tive_ber_coef					64h		
																post_fec_ber_magnitude									post_fec_ber_ coef					68h		

Table 1010 - Physical Layer Statistical Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 6Ch-F4h

Table 1011 - Physical Layer Statistical Counters Fields

Offset	Bits	Name	Description	Access
00h	31:0	time_since_last_clear_high	The time passed since the last counters clear event in msec. [Internal] supported in USR	RO
04h	31:0	time_since_last_clear_low		
08h	31:0	phy_received_bits_high	This counter provides information on the total amount of traffic (bits) received. [Internal] supported in USR	RO
0Ch	31:0	phy_received_bits_low		
10h	31:0	phy_symbol_errors_high	This counter provides information on error bits that were not corrected by FEC correction algorithm or that FEC is not active. [Internal] supported in USR	RO
14h	31:0	phy_symbol_errors_low		
18h	31:0	phy_corrected_bits_high	Corrected bits by FEC engine. [Internal] supported in USR	RO
1Ch	31:0	phy_corrected_bits_low		
20h	31:0	phy_raw_errors_lane0_high	This counter provides information on error bits that were identified on lane 0. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 0 [Internal] RS-FEC : corrected_symbols_lane0 FC-FEC : correcte_blocks_lane0 NO-FEC : Bip errors lane 0 [Internal] supported in USR	RO
24h	31:0	phy_raw_errors_lane0_low		

Table 1011 - Physical Layer Statistical Counters Fields

Offset	Bits	Name	Description	Access
28h	31:0	phy_raw_errors_lane1_high	This counter provides information on error bits that were identified on lane 1. When FEC is enabled this induction corresponds to corrected errors In PRBS test mode, indicates the number of PRBS errors on lane 1 [Internal] RS-FEC : corrected_symbols_lane1 FC-FEC : correcte_blocks_lane1 NO-FEC : Bip errors lane 1 [Internal] supported in USR	RO
2Ch	31:0	phy_raw_errors_lane1_low		
30h	31:0	phy_raw_errors_lane2_high	This counter provides information on error bits that were identified on lane 2. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 2 [Internal] RS-FEC: corrected_symbols_lane2 FC-FEC: correcte_blocks_lane2 NO-FEC: Bip errors lane 2 [Internal] supported in USR	RO
34h	31:0	phy_raw_errors_lane2_low		
38h	31:0	phy_raw_errors_lane3_high	This counter provides information on error bits that were identified on lane 3. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 3 [Internal] RS-FEC: corrected_symbols_lane3 FC-FEC: correcte_blocks_lane3 NO-FEC: Bip errors lane 3 [Internal] supported in USR	RO
3Ch	31:0	phy_raw_errors_lane3_low		

Table 1011 - Physical Layer Statistical Counters Fields

Offset	Bits	Name	Description	Access
40h	31:0	phy_raw_errors_lane4_high	This counter provides information on error bits that were identified on lane 4. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 4 [Internal] RS-FEC: corrected_symbols_lane4 FC-FEC/ZL-FEC: correcte_blocks_lane4 [Internal] supported in USR	RO
44h	31:0	phy_raw_errors_lane4_low		
48h	31:0	phy_raw_errors_lane5_high	This counter provides information on error bits that were identified on lane 5. When FEC is enabled this induction corresponds to corrected errors In PRBS test mode, indicates the number of PRBS errors on lane 5 [Internal] RS-FEC: corrected_symbols_lane5 FC-FEC/ZL-FEC: correcte_blocks_lane5 [Internal] supported in USR	RO
4Ch	31:0	phy_raw_errors_lane5_low		
50h	31:0	phy_raw_errors_lane6_high	This counter provides information on error bits that were identified on lane 6. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 6 [Internal] RS-FEC: corrected_symbols_lane6 FC-FEC/ZL-FEC: correcte_blocks_lane6 [Internal] supported in USR	RO
54h	31:0	phy_raw_errors_lane6_low		
58h	31:0	phy_raw_errors_lane7_high	This counter provides information on error bits that were identified on lane 7. When FEC is enabled this induction corresponds to corrected errors. In PRBS test mode, indicates the number of PRBS errors on lane 7 [Internal] RS-FEC: corrected_symbols_lane7 FC-FEC/ZL-FEC: correcte_blocks_lane7 [Internal] supported in USR	RO
5Ch	31:0	phy_raw_errors_lane7_low		

Table 1011 - Physical Layer Statistical Counters Fields

Offset	Bits	Name	Description	Access
60h	15:8	raw_ber_magnitude	Raw_BER = $raw\_ber\_coef * 10^{(-raw\_ber\_magnitude)}$ [Internal] supported in USR	RO
	3:0	raw_ber_coef		RO
64h	15:8	effective_ber_magnitude	Effective_BER = $effective\_ber\_coef * 10^{(-effective\_ber\_magnitude)}$ [Internal] supported in USR	RO
	3:0	effective_ber_coef		RO
68h	15:8	post_fec_ber_magnitude	post_fec_BER = $post\_fec\_ber\_coef * 10^{(-post\_fec\_ber\_magnitude)}$	RO
	3:0	post_fec_ber_coef		RO

28.19.12.18 PLR Counters Group

Table 1012 - PLR Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
plr_rcv_codes_high																															00h	
plr_rcv_codes_low																															04h	
plr_rcv_code_err_high																															08h	
plr_rcv_code_err_low																															0Ch	
plr_rcv_uncorrectable_code_high																															10h	
plr_rcv_uncorrectable_code_low																															14h	
plr_xmit_codes_high																															18h	
plr_xmit_codes_low																															1Ch	
plr_xmit_retry_codes_high																															20h	
plr_xmit_retry_codes_low																															24h	
plr_xmit_retry_events_high																															28h	
plr_xmit_retry_events_low																															2Ch	
plr_sync_events_high																															30h	
plr_sync_events_low																															34h	
hi_retransmission_rate_high																															38h	
hi_retransmission_rate_low																															3Ch	
plr_xmit_retry_codes_within_t_sec_max_high																															40h	
plr_xmit_retry_codes_within_t_sec_max_low																															44h	
															pre_plr_ber_magnitude												pre_plr_ber_c oef				48h	

Table 1012 - PLR Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																4Ch-F4h

Table 1013 - PLR Counters Fields

Offset	Bits	Name	Description	Access
00h	31:0	plr_rcv_codes_high	Number of received PLR codewords	RO
04h	31:0	plr_rcv_codes_low		
08h	31:0	plr_rcv_code_err_high	The total number of bad codewords received (including uncorrectable).	RO
0Ch	31:0	plr_rcv_code_err_low		
10h	31:0	plr_rcv_uncorrectable_code_high	The number of uncorrectable codewords received	RO
14h	31:0	plr_rcv_uncorrectable_code_low		
18h	31:0	plr_xmit_codes_high	Number of transmitted PLR codewords	RO
1Ch	31:0	plr_xmit_codes_low		
20h	31:0	plr_xmit_retry_codes_high	The total number of codewords retransmitted.	RO
24h	31:0	plr_xmit_retry_codes_low		
28h	31:0	plr_xmit_retry_events_high	The total number of retransmitted event.	RO
2Ch	31:0	plr_xmit_retry_events_low		
30h	31:0	plr_sync_events_high	The number of sync events	RO
34h	31:0	plr_sync_events_low		
38h	31:0	hi_retransmission_rate_high	High retransmission rate The number of the retransmission rate has crossed the retransmission rate threshold	RO
3Ch	31:0	hi_retransmission_rate_low		
40h	31:0	plr_xmit_retry_codes_within_t_sec_max_high	The maximum number of retransmitted events in t sec window	RO
44h	31:0	plr_xmit_retry_codes_within_t_sec_max_low		
48h	15:8	pre_plr_ber_magnitude	pre_plr_BER = $pre\_plr\_ber\_coef * 10^{(-pre\_plr\_ber\_magnitude)}$	RO
	3:0	pre_plr_ber_coef		RO

28.19.12.19 [Internal] RS Histogram Counters Group

Table 1014 - RS-Histograms Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hist[0]																																00h-04h
hist[1]																																08h-0Ch
...																																...
hist[15]																																78h-7Ch
																																80h-F4h

Table 1015 - RS-Histograms Fields

Offset	Bits	Name	Description	Access
00h-7Ch	64	hist[16]	Value of RS-hist	RO

28.19.12.20 [Internal] ZL FEC Counters Group

Table 1016 - ZL FEC Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rewind_counter_high																																00h
rewind_counter_low																																04h
data_rewined_high																																08h
data_rewined_low																																0Ch
pkt_rewined_high																																10h
pkt_rewined_low																																14h
total_rcv_data_high																																18h
total_rcv_data_low																																1Ch
total_rcv_pkt_high																																20h
total_rcv_pkt_low																																24h
stay_in_slow_high																																28h

Table 1016 - ZL FEC Counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
stay_in_slow_low																															2Ch	
																																30h-F4h

Table 1017 - ZL FEC Counters Fields

Offset	Bits	Name	Description	Access
00h	31:0	rewind_counter_high	The number of all rewind events	RO
04h	31:0	rewind_counter_low		
08h	31:0	data_rewined_high	The amount of data that were rewinded	RO
0Ch	31:0	data_rewined_low		
10h	31:0	pkt_rewined_high	The amount of packets that were rewinded	RO
14h	31:0	pkt_rewined_low		
18h	31:0	total_rcv_data_high	The totoal number of received data - both good and rewinded data.	RO
1Ch	31:0	total_rcv_data_low		
20h	31:0	total_rcv_pkt_high	The totoal number of received packets - both good and rewinded data.	RO
24h	31:0	total_rcv_pkt_low		
28h	31:0	stay_in_slow_high	Indication that the ZL FEC entered “stay in slow” (no transition to fast) due to bubbles. [Internal] FW should implemented using one bit. Cleared on port toggling.	RO
2Ch	31:0	stay_in_slow_low		

28.19.13PUDE - Port Up/Down Event

The PUDE register reports the port operational state change of a port. A change in the port’s operational state will generate PUDE only if the change happens after arming the event mechanism (using PAOS register). Any changes to the operational state while the event mechanism is not armed will not be reported.

Table 1018 - PUDE - Port Up/Down Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
swid								local_port												admin_status								oper_status				00h	
																																	04h
																																	08h

Table 1018 - PUDE - Port Up/Down Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h

Table 1019 - PUDE - Port Up/Down Event Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid	For HCA: must ~~be~~ always be 0. Switch partition ID with which to associate the port. Switch partitions are numbered from 0 to 7 inclusively. Switch partition 254 indicates stacking ports. Switch partition 255 indicates all switch partitions. Note: While external ports uses unique local port numbers (and thus swid is redundant), router ports use the same local port number where swid is the only indication for the relevant port.	Index
	23:16	local_port	Local port number.	Index
	11:8	admin_status	Port administrative state (the desired state of the interface): • 0001 - up • 0010 - down by configuration • 0011 - up once - if the port goes up and then down, the operational status should go to “down by port failure” and can only go back up upon explicit command • 0100 - disabled by system	RO
	3:0	oper_status	Port operational state: • 0001 - up • 0010 - down • 0100 - down by port failure (transitioned by the hardware) [Internal] down by port failure state is used to indicate the port was enabled by the admin but moved to down. The device will not let the port up again without software explicit command (UP).	RO

28.19.14 PMPE - Port Module Plug/Unplug Event

The PMPE register reports ~~the module plug and unplug state change of a module~~ any operational status change of a module. A change in the module's state will generate an event only if the change happens after arming the event mechanism. Any changes to the module state while the event mechanism is not armed will not be reported. Software can query the PMPE register for module status.

Table 1020 - PMPE - Port Module Plug/Unplug Event Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
								module																						module_status				00h
																				error_type													04h	
																																08h		
																																0Ch		

Table 1021 - PMPE - Port Module Plug/unplug Event Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
	3:0	module_status	- Port operational state: 1- plugged, enabled 2- unplugged 3- module plugged, error (details in <i>error_type</i>) 4- plugged, disabled	RO
04h	11:8	error_type ¹	Module error details: 0x0: Power_Budget_Exceeded 0x1: Long_Range_for_non_MLNX_cable_or_module 0x2: Bus_stuck - (I2C Data or clock shorted) 0x3: bad_or_unsupported_EEPROM 0x4: Enforce_part_number_list 0x5: unsupported_cable 0x6: High_Temperature 0x7: bad_cable - (Module/Cable is shorted) 0x8: PMD_type_is_not_enabled - (see PMTPS). [internal]0x9: Laster_TEC_failure [internal]0xa: High_current [internal]0xb: High_voltage 0xc: pcie_system_power_slot_Exceeded [internal]0xd: High_power [internal]0xe: Module_state_machine_fault Valid only when (<i>oper_status</i> = 0x3).	RO

1. In case of more than one error, one of the errors will be reported.

28.19.15PMPC - Port Module Present Change

The PMPC register reports ~~the module plug and unplug state change of a module~~ any change in the operational status of the module. A change in the module's state sets the relevant bit in the bit vector. This register can be used for polling of change in cable state. Note that a bit set indicates one or more changes in the status has happened and cannot be used to calculate the actual state of the module. The actual state can be queried using the PMPE register.

Table 1022 - PMPC – Port Module Present Change Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
module_state_updated																																0h-1Ch

Table 1023 - PMPC – Port Module Present Change Register Fields

Offset	Bits	Name	Description	Access
0h-1Ch	256	module_state_updated	On read operation: A bit vector where each bit represents a module (bit 0 represents module 0 etc). A set bit indicates one or more changes in the module state (plugged/unplugged) has happened since the last time the bit was cleared. The default is 1 in all relevant bits. On write operation: Used to clear the module_status_updated bits. Writing a set bit clears the relevant bit in the module_status_updated vector. Cleared bits in this field preserve the value of the relevant bit in the bit vector. The software cannot set a bit in the module_status_updated vector. Note: To prevent a case where an update is missed by the software, it is recommended to use the flow below: • Read the module_status_updated vector • Find a bit(s) relevant to a module(s) the software would like to handle • Clear the relevant bit(s) • Handle the module (e.g. read the module parameters using MCIA register) Read module_status_updated again to make sure the status did not change after reading the cable information.	RW

28.19.16 PLIB - Port Local port to InfiniBand and label port mapping Register

The PLIB register performs mapping from Local Port into InfiniBand Port.

Table 1024 - PLIB - Port Local-Port to IB Port Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1025 - PLIB - Port Local-Port to IB Port Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	7:0	ib_port	In IB port: InfiniBand port remapping for local_port In Ethernet port: Label port remapping for local_port Note: ib_port number can only be updated when a port admin state is DISABLED.	RW
04h	3:0	split_num	Valid only for Ethernet Switches. Label split mapping for local_port	RW

28.19.17PFSC - Port Forward Switching Control

The PFSC register enable the enforcement of S&F operation per egress port.

PFSC operational status will be update only after port toggling.

Table 1026 - PFSC - Port Forward Switching Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1027 - PFSC - Port Forward Switching Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
04h	3:0	fwd_admin	Administratively configured of Forward switching for Egress: 0 - Allow cut-through forwarding 1 - Always Store and Forward	RW
08h	3:0	fwd_oper	Operational Egress Forward switching: 0 - Cut-Through forwarding operation is allowed 1 - Only Store and Forward operation	RO

28.19.18PMMP - Port Module Memory Map Properties Register

This register enables to override the cable/module memory map advertisement.

The register can be set only when the module is disabled by PMAOS admin status.

Note: Overriding a cable memory map advertising does not impact its physical properties. A mismatch between the configured properties and the cable/module capabilities can result in link up failure or performance degradation.

[Internal] NOTE: This Register DOES NOT override Module Locking as defined in MCAS - Locking policy is always enforced!

Table 1028 - PMMP - Port Module Memory Map Properties Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
								module																										00h
eeprom_override_mask																eeprom_override																04h		
qsfp_cable_protocol_technology																																08h-10h		
sfp_cable_protocol_technology																																14h-1Ch		
																									cable_length								20h	
attenuation_25g								attenuation_12g								attenuation_7g								attenuation_5g								24h		
																										module_identifier								28h

Table 1029 - PMMP - Port Module Memory Map Properties Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
04h	31:16	eeprom_override_mask	Write Mask bit (negative polarity): 0 - allow write 1 - ignore write On write commands, indicates which of the bits from <i>eeprom_override</i> field are updated. Reserved on read operations.	WO
	15:0	eeprom_override ¹	Override/ignore EEPROM advertisement properties bitmask: Bit 0 - Override cable protocols and technology for QSFP Bit 1 - Override cable protocols and technology for SFP Bit 2- Ignore Power Class (set high power) Bit 3- Override Cable Length Bit 4- Override Attenuation [Internal] Bit 5- Override module identifier Bit 8 - Set Module to Low Power	RW
08h-10h	96	qsfp_cable_protocol_technology	Valid only when the protocols and technology is overridden as indicated by <i>eeprom_override</i> See Table 1030, “PMMP - QSFP Protocol Bytes Override Layout,” on page 1232	RW
14h-1Ch	96	sfp_cable_protocol_technology	Valid only when the protocols and technology is overridden as indicated by <i>eeprom_override</i> See Table 1032, “PMMP - SFP Protocol Bytes Override Layout,” on page 1234	RW
20h	7:0	cable_length	Length of cable assembly, units of 1 m. Valid only when the cable length is overridden as indicated by <i>eeprom_override</i> .	RW
24h	31:24	attenuation_25g	Valid only when the attenuation is overridden as indicated by <i>eeprom_override</i> . Attenuation - total channel attenuation @ 25GHz in db.	RW
	23:16	attenuation_12g	Valid only when the attenuation is overridden as indicated by <i>eeprom_override</i> . Attenuation - total channel attenuation @ 12GHz in db.	RW
	15:8	attenuation_7g	Valid only when the attenuation is overridden as indicated by <i>eeprom_override</i> . Attenuation - total channel attenuation @ 7GHz in db.	RW
	7:0	attenuation_5g	Valid only when the attenuation is overridden as indicated by <i>eeprom_override</i> . Attenuation - total channel attenuation @ 5GHz in db.	RW

Table 1029 - PMMP - Port Module Memory Map Properties Register Fields

Offset	Bits	Name	Description	Access
28h	7:0	module_identifier	Valid only when the module identifier is overridden as indicated by <i>eeeprom_override</i> . 0x3 - SFP+/SFP28 0xD - QSFP+ 0x11 - QSFP28 0x18 - QSFP-DD 0x19 - OSFP 0x1A - SFP-DD	RW

1. Overriding of both SFP and QSFP protocols and technology is not allowed.

28.19.18.1 QSFP Protocol Bytes Override

Table 1030 - PMMP - QSFP Protocol Bytes Override Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1031 - PMMP - QSFP Protocol Bytes Override Fields

Offset	Bits	Name	Description	Access
00h	15:8	cable_breakout	Override of byte113 per SFF-8636: Bit 6-4 (Far End): 0 - Far end is unspecified 1 - Cable with single far end with 4 channels implemented, or separable module with 4-channel connector 2 - Cable with single far end with 2 channels implemented, or separable module with 2-channel connector 3 - Cable with single far end with 1 channel implemented, or separable module with 1-channel connector 4 - 4 far ends with 1 channel implemented in each (i.e. 4x1 break out) 5 - 2 far ends with 2 channels implemented in each (i.e. 2x2 break out) 6 - 2 far ends with 1 channel implemented in each (i.e. 2x1 break out) Bit 3-0 (Near End) Bit 0 - Channel 1 implementation Bit 1 - Channel 2 implementation Bit 2 - Channel 3 implementation Bit 3 - Channel 4 implementation for each channel: 0 - Channel implemented 1 - Channel not - implemented	RW
	7:0	ethernet_compliance_code	Ethernet Compliance Codes bit mask (10/40G/100G) Override byte131 per SFF-8636: Bit 7 - Extended Specification Compliance valid Bit 6 - 10GBASE-LRM Bit 5 - 10GBASE-LR Bit 4 - 10GBASE-SR Bit 3 - 40GBASE-CR4 Bit 2 - 40GBASE-SR4 Bit 1 - 40GBASE-LR4 Bit 0 - 40G Active Cable (XLPPI)	RW

Table 1031 - PMMP - QSFP Protocol Bytes Override Fields

Offset	Bits	Name	Description	Access
04h	15:8	ext_ethernet_compliance_code	Extended Specification Compliance Codes Override byte192 per SFF-8636: 0 - Unspecified 1 - 100G AOC (Active Optical Cable) or 25GAUI C2M AOC with FEC 2 - 100GBASE-SR4 or 25GBASE-SR 3 - 100GBASE-LR4 or 25GBASE-LR 4 - 100GBASE-ER4 or 25GBASE-ER 5 - 100GBASE-SR10 6 - 100G CWDM4 7 - 100G PSM4 Parallel SMF 8 - 100G ACC (Active Copper Cable) or 25GAUI C2M ACC. with FEC 11 - 100GBASE-CR4 or 25GBASE-CR CA-L 12 - 25GBASE-CR CA-S 13 - 25GBASE-CR CA-N 16 - 40GBASE-ER4 17 - 4 x 10GBASE-SR 18 - 40G PSM4 Parallel SMF 19 - G959.1 profile P1I1-2D1 (10709 MBd, 2km, 1310nm SM) 20 - G959.1 profile P1S1-2D2 (10709 MBd, 40km, 1550nm SM) 21 - G959.1 profile P1L1-2D2 (10709 MBd, 80km, 1550nm SM) 22 - 10GBASE-T with SFI electrical interface 23 - 100G CLR4 24 - 100G AOC or 25GAUI C2M AOC. No FEC 25 - 100G ACC or 25GAUI C2M ACC. No FEC	RW
	7:0	giga_ethernet_compliance_code	Gigabit Ethernet Compliance Codes Override byte134 per SFF-8636: Bit 3 - 1000BASE-T Bit 2 - 1000BASE-CX Bit 1 - 1000BASE-LX Bit 0 - 1000BASE-SX	RW

28.19.18.2 SFP Protocol Bytes Override

Table 1032 - PMMP - SFP Protocol Bytes Override Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								bit_rate								cable_technology								tengig_ethernet_compliance_code								00h
																ext_ethernet_compliance_code								ethernet_compliance_code								04h
																																08h

Table 1033 - PMMP - SFP Protocol Bytes Override Fields

Offset	Bits	Name	Description	Access
00h	23:16	bit_rate	Override byte12 per SFF-8472 The nominal bit (signaling) rate (BR, nominal) is specified in units of 100 MBd, rounded off to the nearest 100 MBd. A value of FFh indicates the bit rate is greater than 25.0Gb/s	RW
	15:8	cable_technology	SFP+ Cable Technology Override byte8 per SFF-8472: Bit 3 - Active Cable Bit 2 - Passive Cable	RW
	7:0	tengig_ethernet_compliance_code	10G Ethernet Compliance Codes Override byte3 per SFF-8472: Bit 7 - 10G Base-ER Bit 6 - 10G Base-LRM Bit 5 - 10G Base-LR Bit 4 - 10G Base-SR	RW

Table 1033 - PMMP - SFP Protocol Bytes Override Fields

Offset	Bits	Name	Description	Access
04h	15:8	ext_ethernet_compliance_code	Extended Specification Compliance Codes (25GE) Override byte36 per SFF-8472: 0 - Unspecified 1 - 100G AOC (Active Optical Cable) or 25GAUI C2M AOC with FEC 2 - 100GBASE-SR4 or 25GBASE-SR 3 - 100GBASE-LR4 or 25GBASE-LR 4 - 100GBASE-ER4 or 25GBASE-ER 5 - 100GBASE-SR10 6 - 100G CWDM4 7 - 100G PSM4 Parallel SMF 8 - 100G ACC (Active Copper Cable) or 25GAUI C2M ACC. with FEC 11 - 100GBASE-CR4 or 25GBASE-CR CA-L 12 - 25GBASE-CR CA-S 13 - 25GBASE-CR CA-N 16 - 40GBASE-ER4 17 - 4 x 10GBASE-SR 18 - 40G PSM4 Parallel SMF 19 - G959.1 profile P1I1-2D1 (10709 MBd, 2km, 1310nm SM) 20 - G959.1 profile P1S1-2D2 (10709 MBd, 40km, 1550nm SM) 21 - G959.1 profile P1L1-2D2 (10709 MBd, 80km, 1550nm SM) 22 - 10GBASE-T with SFI electrical interface 23 - 100G CLR4 24 - 100G AOC or 25GAUI C2M AOC. No FEC 25 - 100G ACC or 25GAUI C2M ACC. No FEC	RW
	7:0	ethernet_compliance_code	Ethernet Compliance Codes Override byte6 per SFF-8472: Bit 7 - BASE-PX Bit 6 - BASE-BX10 Bit 5 - 100BASE-FX Bit 4 - 100BASE-LX/LX10 Bit 3 - 1000BASE-T Bit 2 - 1000BASE-CX Bit 1 - 1000BASE-LX Bit 0 - 1000BASE-SX	RW

28.19.19PMCR - Port Module Control Register

This register enables to override module configurations performed by the port during link bring up.

The register can be set only when the port is down by PAOS admin status.

Note: For some properties the port auto-mode does not modify the cable/module property and relies on the default value. If that property is modified using override / an external write, the auto behavior might result in a different behavior.

Table 1034 - PMCR - Port Module Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
						cs_sel	local_port																										00h
						tx_disable_override_value								tx_disable_override_cntl						cdr_override_value												cdr_override_cntl	04h
						rx_loss_override_admin								rx_loss_override_cntl						rx_amp_override_value												rx_amp_override_cntl	08h
																				rx_emp_override_value												rx_emp_override_cntl	0Ch
						tx_adaptive_override_cntl														tx_equ_override_value												tx_equ_override_cntl	10h

Table 1034 - PMCR - Port Module Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																				rs_override_value												rs_override_cntl	14h
						si_override_cntl														ap_sel_override_value												ap_sel_override_cntl	18h

Table 1035 - PMCR - Port Module Control Register Fields

Offset	Bits	Name	Description	Access
00h	25:24	cs_sel	Valid only for CMIS (QSFP-DD / SFP-DD / OSFP) compliant modules. If Control Set 1 not implemented or non CMIS module ignore field. 0 - Auto (default) 1 - Use Control Set 0 2 - Use Control Set 1	Index
	23:16	local_port	Local port number.	Index
04h	25:24	tx_disable_override_value	Valid only in case TX_DISABLE is configured to override mode by tx_disable_override_cntl : 0 - TX_DISABLE is low (i.e. TX is enabled) [Internal] 1 - TX_DISABLE is high (i.e. TX is disabled)	RW
	17:16	tx_disable_override_cntl	In HCA, valid based on PCAM feature capabilities. Module TX_DISABLE override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW
	11:8	cdr_override_value	Valid only in case CDR is configured to override mode by cdr_override_cntl : 0 - RX OFF, TX OFF 1 - RX ON , TX OFF 2 - RX OFF, TX ON 3 - RX ON, TX ON	RW
	1:0	cdr_override_cntl	Module CDR override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW

Table 1035 - PMCR - Port Module Control Register Fields

Offset	Bits	Name	Description	Access
08h	25:24	rx_los_override_admin	Module RX_LOS override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW
	17:16	rx_los_override_cntl	Valid only in case RX_LOS is configured to override mode by rx_los_override_admin	RW
08h	11:8	rx_amp_override_value	Valid only in case RX amplitude is configured to override mode by rx_amp_override_cntl	RW
	1:0	rx_amp_override_cntl	Module RX amplitude override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW
0Ch	11:8	rx_emp_override_value	Valid only in case RX emphasis is configured to override mode by rx_emp_override_cntl	RW
	1:0	rx_emp_override_cntl	Module RX emphasis override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW
10h	25:24	tx_adaptive_override_ctrl	Module adaptive tuning flow override control: 0 - Auto (default) 1 - Keep the module configuration	RW
	11:8	tx_equ_override_value	Valid only in case TX equalization is configured to override mode by tx_equ_override_cntl	RW
	1:0	tx_equ_override_cntl	Module TX equalization override control: 0 - Auto (default) 1 - Keep the module configuration 2 - Override	RW
14h	11:8	rs_override_value	Valid only in case Rate Select is configured to override mode by rs_override_cntl : For QSFP modules: Bits 1,0 RX , Bits 3,2 TX ; 0 - MSB Low, LSB Low 1 - MSB Low , LSB High 2 - MSB High, LSB Low 3 - MSB High , LSB High For SFP module Bit 0 is RS0 Bit 2 is RS1. Bits 1,3 are Reserved	RW
	1:0	rs_override_cntl	Module Rate Select override control: 0 - Auto (default) 1 - Use Module's Rate Select value 2 - Override	RW

Table 1035 - PMCR - Port Module Control Register Fields

Offset	Bits	Name	Description	Access
18h	25:24	si_override_cntrl	Valid only for CMIS (QSFP-DD / SFP-DD / OSFP) compliant modules. If not CMIS module ignore field. Module Signal Integrity override control: 0 - Auto (default) 1 - Override with Module's Default applications SI values 2 - Override with Hosts SI values	RW
	11:8	ap_sel_override_value	Application Select from the applications list advertised by the module. Up to 15 applications available. Values range from 1 to 15. Note: application can be set with default SI values defined by the module or SI defined by the host	RW
	1:0	ap_sel_override_cntrl	Module Application select override control: 0 - Auto (default) 1 - Use Module's Default application value 2 - Override	RW

28.19.20 PGMR - Port And Group Mapping Register

The register enables to read or set the relation between groups and local ports.

Table 1036 - PGMR- Port and Group Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status				version				local_port								pnat				group								pg_sel	00h				
																								group_of_port								04h	
ports_mapping_of_group																																08h-14h	

Table 1037 - PGMR - Port and Group Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 16nm products	RO
	23:16	local_port	Local port number	Index
	15:14	pnat	Port number access type. Determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number 3 - Out of band / PCI	Index
	11:8	group	Group number	Index
	0	pg_sel	Selection between an operation on a local port or a group, for 'set' and 'get' operations. 0 - local port (<i>group</i> field is neglected even if set) 1 - group (<i>local_port</i> field is neglected even if set)	OP
04h	7:4	group_of_port	The corresponding group number for the <i>local_port</i>	RW
08h-14h	128	ports_mapping_of_group	A bit map of all available local ports in the system (each bit represents a local port). The corresponding bits which represent the local ports for the specific group in the <i>group</i> field, are set to '1'. If set to '0' - not in that specific group or not available.	RW

28.19.21 PMDR - Port Mapping Definitions Register

The register enables to query the relation between port mapping on the logical and physical domains.

Table 1038 - PMDR- Port Mapping Definitions Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
status				version				local_port								pnat		gb_valid		mcm_tile_valid														00h	
pport									module									cluster									clp_4x			clp_1x		04h			
				split								swid												ib_port								08h			
	label_port_query							local_port_query								usr_sd_4x_tx		usr_sd_4x_rx		sd_4x_tx		sd_4x_rx		sd_lane_mask_tx				sd_lane_mask_rx				0Ch			
								mcm_tile_num																			gearbox_die_num								10h
																																	14h-1Ch		

Table 1039 - PMDR - Port Mapping Definitions Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid 2 - Bad Param (Values are not ready yet - during polling) 3 - Invalid index of local_port or label_port	RO
	27:24	version	0 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number 3 - Out of band / PCI	Index
	13	gb_valid	0 if there is no gearbox arch	RO
	12	mcm_tile_valid	0 if there is no MCM tile arch	RO

Table 1039 - PMDR - Port Mapping Definitions Register Fields

Offset	Bits	Name	Description	Access
04h	31:24	pport		RO
	22:16	module		RO
	14:8	cluster	Cluster number	RO
	5:4	clp_4x	Logical cluster port 1x	RO
	1:0	clp_1x	Logical cluster port 4x	RO
08h	26:24	split	Number of ports the label port is split to. 0 - split to 1 1 - split to 2 2 - split to 4	RO
	18:16	swid		RO
	6:0	ib_port		RO
0Ch	30:24	label_port_query	Label port index for the local port chosen (in case <i>pnat</i> field is '0').	RO
	23:16	local_port_query	Local port index for the label port chosen (in case <i>pnat</i> field is '1').	RO
	15:14	usr_sd_4x_tx	USR serdes 4x TX 00b - no USR serdes 01b - serdes physical tx maps to 0-3 10b - serdes physical tx maps to 4-7	
	13:12	usr_sd_4x_rx	USR serdes 4x RX 00b - no USR serdes 01b - serdes physical rx maps to 0-3 10b - serdes physical rx maps to 4-7	
	11:10	sd_4x_tx	LR serdes 4x TX	RO
	9:8	sd_4x_rx	LR serdes 4x RX	RO
	7:4	sd_lane_mask_tx		RO
	3:0	sd_lane_mask_rx		RO
10h	23:16	mcm_tile_num	Number of the MCM Tile the <i>local_port</i> is related to. valid only when mcm_tile_valid is 1	RO
	6:0	gearbox_die_num	Number of the Gearbox the <i>local_port</i> is related to. valid only when gb_valid is 1	RO

28.19.22[Internal] PMMIC - Port Module Management Initialization Checker Register

This register enables to override entire cable/module memory map pages for module init flows validation purpose.

The register can be set only when the module is disabled by PMAOS admin status.

Note: Overriding a cable memory map while in mission mode (desire cable is plugged) may cause mismatch between the configured properties and the cable/module capabilities can result in link up failure or performance degradation.

Table 1040 - PMMIC - Port Module Management Initialization Checker Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
last	unlock																			changed_ - byte_index												00h
																																04h
																																08h
																																0Ch
																																10h-8Ch

Table 1041 - PMMIC - Port Module Management Initialization Checker Register Fields

Offset	Bits	Name	Description	Access
00h	31	last	Last page in “data” array	RW
	30	unlock	When set, EEPROM simulation is ceased on this module	OP
	23:16	module	Module number.	Index
	11:8	changed_byte_index	Index between 0 to <i>num_of_changed_value</i> set which “changed” bytes to show in <i>bank_value</i> , <i>page_value</i> , <i>byte_addr</i> and <i>byte_value</i> .	Index
	7:0	status	Module initialization flow final result: 0-Success – (module ready) 1-In progress 2-Link down: unknown_identifier 3-Link down: high_power_not_allowed 4-Link down: cable_stamping / LR-non-mlnx cable 5-Link down: com codes mismatch	RO

Table 1041 - PMMIC - Port Module Management Initialization Checker Register Fields

Offset	Bits	Name	Description	Access
04h	31:28	qsa_type	Used for checking QSA initialization flows: 0 - NO QSA 1 - QSA10 (no memory map) 2 - QSA 28 3 - QSA 56 4 - DDQSA 5- OQSA	RW
	23:16	page_number	page number	RW
	8	lower	Valid only when <i>page_number</i> = 0 When set, the data is referred to lower page 0 bytes 0-127	RW
	7:0	i2c_slave_addr	I2C module/device address	RW
08h	3:0	num_of_changed_bytes	Number of bytes that were changed by Module Management Init Flow.	RO
0Ch	31:24	byte_bank	Referring to <i>changed_byte_index</i> byte bank that was written during the module initialization flow	RO
	23:16	byte_page	Referring to <i>changed_byte_index</i> byte page that was written during the module initialization flow	RO
	15:8	byte_addr	Referring to <i>changed_byte_index</i> byte addr that was written during the module initialization flow	RO
	7:0	byte_value	Referring to <i>changed_byte_index</i> byte value that was written during the module initialization flow	RO
10h-8Ch	1024	data	EEPROM page data	RW

28.19.23 PEPC - Port External Phy Control Register

This register enables to manage external PHY properties.

Table 1042 - PEPC - Port External Phy Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch

Table 1043 - PEPC - Port External Phy Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
00h	3:0	status	0x0 - Good 0x1 - No support of the required configuration 0x2 - Unknown External PHY 0xF - No External PHY	RO
04h	16	twisted_pair_an	AN operation mode: 0 - Normal operation - AN is determine by MDI 1 - Disable AN	RW
04h	8	twisted_pair_force_mode	twisted_pair operational mode in case of AN disabled: 0x0 - Master mode 0x1 - Slave mode	RW
	1:0	twisted_pair_an_mode	twisted_pair operational mode in case of AN enabled: 0x0 - auto Master/Slave mode for AN links. 0x1 - Master mode 0x2 - Slave mode	RW

28.19.24PPTB - Port Prio To Buffer

The PPTB register configures the Prio to the Buffer table.

Table 1044 - PPTB - Port Prio To Buffer Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
		mm						local_port								pnat							cm	um	pm								00h
prio7buff				prio6buff				prio5buff				prio4buff				prio3buff				prio2buff				prio1buff				prio0buff				04h	
pm_msb																									ctrl_buff				untagged_buff				08h

Table 1045 - PPTB - Port Prio To Buffer Register Fields

Offset	Bits	Name	Description	Access
00h	29:28	mm ¹	Mapping mode 0 - Mapping both Unicast and Multicast packets to the same buffer In get operation, 0 means Unicast only. 1 - Mapping Unicast only 2 - Mapping Multicast only This field is reserved for HCA devices	Index
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - Label port number 3 - Out of band / PCI	Index
	9	cm	ctrl_buff mask - enables update of the ctrl_buff field. NOTE: This field is ignored by the device, and control traffic is mapped to a pre-defined buffer (buffer - 9). In some old implementations (on SwitchX and SwitchX-2) software is required to use the value 9 in this field, later devices ignore this field. Reserved for HCA devices	WO
	8	um ²	untagged_buff mask - enables update of the untagged_buff field. Reserved for Spectrum. Reserved for HCA devices.	WO
	7:0	pm	prioX_buff mask - enables update of the prioX_buff field. Bit <i>i</i> is a flag for updating the mapping for switch priority <i>i</i> .	WO

Table 1045 - PPTB - Port Prio To Buffer Register Fields

Offset	Bits	Name	Description	Access
04h	31:28	prio7buff	Mapping of Switch Priority of <X> to one of the allocated receive port buffers. <X> = 0..7 Number of supported Switch Priorities is defined in TBD. TODO: Add link	RW
	27:24	prio6buff		RW
	23:20	prio5buff		RW
	19:16	prio4buff		RW
	15:12	prio3buff		RW
	11:8	prio2buff		RW
	7:4	prio1buff		RW
	3:0	prio0buff		RW
08h	31:24	pm_msb	prioX_buff mask - enables update of the prioX_buff field. Bit <i>i</i> is a flag for updating the mapping for switch priority <i>i</i> +8. Reserved for HCA device	WO
	7:4	ctrl_buff	Mapping of control frames to one of the allocated receive port buffers. NOTE: This field is ignored by the device, and control traffic is mapped to a pre-defined buffer (buffer - 9). In some old implementations (on SwitchX and SwitchX-2), software is required to use the value 9 in this field, later devices ignore this field. Reserved for HCA devices	RW
	3:0	untagged_buff	Mapping of untagged frames to one of the allocated receive port buffers. NOTE: In SwitchX/-2, this field must be mapped to a buffer 8. [Internal] For SwitchX/2 buffer 8 can only be used by untagged packets. Reserved for Spectrum. Spectrum maps untagged packets based on the Switch Priority. Reserved for HCA devices	RW

1. SwitchX and SwitchX-2 support the first mode only (mapping of both Unicast and Multicast to the same buffer).
2. In Spectrum, untagged packets are mapped based on the default Switch Priority and do not use this field. This field is reserved in Spectrum.

[Internal] The register is mapping untagged to buffer while HW map untagged to Prio and then Prio to buffer. This is why we ask to map untagged to a buffer already used by some other prio and FW will do the math. FW should enforce mapping of untagged to a buffer used by at least one of the Prios.

[Internal] We are now adding the special buffer allocation for management packets. Once the ECO is done, I will add the extra control.

[Internal] HCA support: The FW default behavior is to update Prio2Buffer mapping based on PFCC only (i.e. set this register automatically). Once PPTB or PBMC touched, FW is aware that higher layer (driver) is controlling the Prio2Buffer mapping and isn't handling this register automatically any more. FW return to default behavior (i.e. automatically mode) upon FLR or tear-down.

28.19.25PBMC - Port Buffer Management Control Register

The PBMC register configures and retrieves the port packet buffer allocation for different Prios, and the Pause threshold management.

Table 1046 - PBMC - Port Buffer Management Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port								pnat																	00h
xoff_timer_value																xoff_refresh																04h	
									fullness_threshold								port_buffer_size																08h
buffer[0]																																0Ch-10h	
buffer[1]																																14h-18h	
buffer[2]																																1Ch-20h	
...																																...	
buffer[9]																																54h-58h	
																																	5Ch-60h
port_shared_buffer																																64h-68h	

Table 1047 - PBMC - Port Buffer Management Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0: Local_port_number 1: Label_port_number 3: Out_of_band_or_PCI	Index
04h	31:16	xoff_timer_value	When device generates a pause frame, it uses this value as the pause timer. The pause timer is measured in units of pause "quanta", where each unit is equal to 512 bit times.	RW
	15:0	xoff_refresh	The time before a new pause frame should be sent to refresh the pause state. Using the same units as xoff_timer_value above.	RW

Table 1047 - PBMC - Port Buffer Management Control Register Fields

Offset	Bits	Name	Description	Access
08h	22:16	fullness_threshold	Almost full threshold configuration. Percentage of buffer fullness which indicates “almost full” occurrence Range: 0-100% Reserved for Switches.	RW
	15:0	port_buffer_size	Total packet buffer array available for the port. The sum of buffer array allocated to bufferX must not exceed port_buffer_size. ~~In Spectrum this field is reserved.~~ ~~TODO: add referral to the place that configures this value! Units are represented in cells (See TBD Appendix G for how many bytes in cell).~~ ~~[Internal] For SwitchX/-2 the PRM shows 64 byte chunks while the HW is 72 byte chunks, and FW handles the conversion. In Spectrum cell is 96 bytes (HW and PRM).~~ In Spectrum and above- Max supported Headroom buffer for port. [internal] Granularity?	RO
0Ch-58h	64	buffer[10] (See Table 1048, “bufferX Register Layout,” on page 1250)	Configuring per-buffer parameters. i=0..9	RW
64h-68h	64	port_shared_buffer	Configuring port shared buffer parameters. Using the same layout as in BufferX, for details see Table 1048, “bufferX Register Layout,” on page 1250. Not supported in SwitchX and SwitchX-2 Valid for HCA when pbmc_port_shared_buffer is set in PCAM.	RW

Table 1048 - bufferX Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
						lossy	epsb									size											00h					
xoff_threshold																xon_threshold											04h					

Table 1049 - bufferX Register Layout

Offset	Bits	Name	Description	Access
00h	25	lossy	The field indicates if the buffer is lossy. 0: Lossless 1: Lossy The field is reserved for Switch-X/-2. For <i>port_shared_buffer</i> , this field is reserved. NOTE: This field does NOT affect the pause control. Pause control is configured using PFCC register.	RW
	24	epsb	Eligible for Port Shared buffer. If <i>epsb</i> is set, packets assigned to bufferX are allowed to insert the port shared buffer. For <i>port_shared_buffer</i> , this field is reserved. When <i>lossy</i> ==1 this field is reserved. The field is reserved for Switch-X/-2.	RW
	15:0	size	The part of the packet buffer array is allocated for the specific buffer. Units are represented in cells. [Internal] Need to check NIC/HCA units.	RW
04h	31:16	xoff_threshold	Once the amount of data in the buffer goes above this value, device starts sending Pause frames for all Prios associated with the buffer. Units are represented in cells. In Spectrum <i>xoff_threshold</i> is reserved for buffer[9]. For Spectrum, if <i>lossy</i> is set on, this field is reserved.	RW
	15:0	xon_threshold	When the amount of data in the buffer goes below this value, device stops sending pause frames for the Prios associated with the buffer. Units are represented in cells. In Spectrum <i>xon_threshold</i> is reserved for buffer[9]. For Spectrum, if <i>lossy</i> is set on, this field is reserved.	RW

Note: Buffers 0 - 9 allocation is under the below restrictions:

- In SwitchX/X2 - On global pause mode, only a single buffer (buffer 1) can be configured. All traffic will go to buffer 1.
- In SwitchX/X2 - Buffer 0 must be used for lossy traffic (no Pause should be used)
- Buffers 8 and 9 can only be used for untagged and control traffic.
- In Spectrum buffer 9 is not subject to flow control.

[Internal] HCA support: The FW default behavior is to update Buffer Allocation and Xon/Xoff thresholds based on PFCC only (i.e. set this register automatically). Once PPTB or PBMC touched, FW is aware that higher layer (driver) is controlling the Buffer Allocation and Xon/Xoff thresholds and isn't handling this register automatically any more. FW return to default behavior (i.e. automatically mode) upon FLR or teardown.

28.19.26PBSR - Port Buffer Status Register

The PBSR register provides information regarding the buffers usage.

Table 1050 - PBSR - Port Buffer Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port																								buffer_type	00h
																																	04h
clear_wm																used_shared_headroom_buffer																08h	
stat_buffer[0]																0Ch-10h																	
stat_buffer[1]																14h-18h																	
stat_buffer[2]																1Ch-20h																	
...																...																	
stat_buffer[9]																54h-58h																	
																																5Ch-60h	

Table 1051 - PBSR - Port Buffer Status Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
00h	1:0	buffer_type	Valid only for 8x port setting (see PMLP): 0x1 - Master buffer 0x2 - Slave buffer	Index
08h	31	clear_wm	Clear watermarks for all PGs	OP
	15:0	used_shared_headroom_buffer	Number of currently used shared headroom buffer cells.	RO
0Ch-58h	64	stat_buffer[10] (Table 1052, "stat_bufferX Register Layout," on page 1253)	status per-buffer parameters. i=0..9	

Table 1052 - stat_bufferX Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																watermark															00h	
																used_buffer															04h	

Table 1053 - stat_bufferX Register Layout

Offset	Bits	Name	Description	Access
00h	15:0	watermark	Watermark in cells (max buffer usage since last clear)	RO
04h	15:0	used_buffer	Number of used buffer cells.	RO

28.19.27PSPA - Port Switch Partition Allocation

This register controls the association of a port with a switch partition, and enables configuring ports as stacking ports.

NOTE: When a local port is disabled by PSPA, this port cannot be configured by any access register (exclude PSPA and PMLP). SET/GET operation of any access register other than PSAP and PMLP on unmapped port will result in BAD returned status.

Table 1054 - PSPA - Port Switch Partition Allocation Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
swid								local_port								sub_port																00h
																																04h

Table 1055 - PSPA - Port Switch Partition Allocation Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	swid	Switch partition ID with which to associate {port, sub_port}. Switch partitions are numbered from 0 to 7 inclusively. Switch partition 254 indicates stacking port. Switch partition 255 indicates a disabled port.	RW
	23:16	local_port	Local port number.	Index
	15:8	sub_port	Reserved for Spectrum/Switch-IB/2. Virtual port within the physical port. 0 - when virtual ports are not enabled on the port.	Index

28.19.28 PELC - Port Extended Link Capabilities

Valid only for SwitchX/2.

The PELC register configures and reports the port extended capabilities. The register enables access to one of the capabilities as defined by the *op* field. Note that changes to the link extended capabilities while the link is up only take effect after link retrain.

Table 1056 - PELC - Port Extended Link Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
op				mlpn_status				local_port																								00h
op_admin								op_capability								op_request								op_active								04h
admin																																08-0Ch
capability																																10-14h
request																																18-1Ch
active																																20-24h
peer_support																																28-2Ch
peer_request																																30-34h

Table 1057 - PELC - Port Extended Link Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	op	Operation 0 - FEC control 1 - LLR control	Index
	27:24	mlpn_status	Mellanox Link Properties Negotiation (MLPN) Status. Indicates the status of the MLPN machine. For state encoding refer to Section 28.19.29, “PMLPN - Port Mellanox Link Protocol Negotiation,” on page 1259	RO
	23:16	local_port	Local port number	Index

Table 1057 - PELC - Port Extended Link Capabilities Register Fields

Offset	Bits	Name	Description	Access
04h	31:24	op_admin	Operation specific administratively enabled properties. Properties of the operation: LLR / FEC administratively enabled by software. HW further negotiates the enabled properties with the remote end of the link. Note: When updating the <i>admin</i> field while the link is up, the changes only affect the link after link retrain. To retrain the link, use PAOS register When Operation = LLR Bit 0 = LL-LLR mode Bits 1-7 Reserved.	RW
	23:16	op_capability	Operation specifies properties capabilities. Same encoding as op_admin	RO
	15:8	op_request	Operation specifies properties request. Same encoding as op_admin	RW
	7:0	op_active	Operation specifies properties active. Same encoding as op_admin	RO

Table 1057 - PELC - Port Extended Link Capabilities Register Fields

Offset	Bits	Name	Description	Access
08h-0Ch	64	admin	Link administratively enabled capabilities by software. HW further negotiates the enabled capabilities with the remote end of the link. Note: When updating the <i>admin</i> field while the link is up, the changes only affect the link after link retrain. To retrain the link, use PAOS register. Ethernet FEC (bitmask) using admin[31:0] <i>[Internal]</i> Bit 9- 56GBase KX4 <i>[Internal]</i> Bit 8 - 56GBase KR4 Bit 7 - 40GBase KR4 Bit 6 - 40GBase CR4 <i>[internal]</i> Bit 5 - 20GBase-KR2 Bit 4 - 10GBase KR Bit 3 - 10GBase KX4 Bit 2 - 10GBase-CX4, XAUI Bit 1 - 1000Base KX Bit 0 - 1000Base-CX, SGMII IB FEC (bitmask) using admin[63:32] ib_link_speed <= ib_proto_capability[7:0] Bit 0 - SDR Bit 1 - DDR Bit 2 - QDR Bit 3 - FDR10 Bit 4 - FDR Bit 5 - EDR Bit 6 - HDR <i>[Internal]</i> Bit 7 - NDR	RW
10h-14h	64	capability	FEC capability supported (bitmask). Same encoding as admin field.	RO
18h-1Ch	64	request	FEC request (bitmask). Same encoding as admin field. Note: When updating the <i>admin</i> field while the link is up, the changes only affect the link after link retrain. To retrain the link, use PAOS register.	RW
20h-24h	64	active	FEC operational mode. Same encoding as admin field.	RO
28h-2Ch	64	peer_support	Link peer capabilities. The same encoding as mlp_capability fields. This field is valid only when the <i>mlpn_status</i> indicates successful negotiation.	RO

Table 1057 - PELC - Port Extended Link Capabilities Register Fields

Offset	Bits	Name	Description	Access
30h-34h	64	peer_request	Link peer request. The same encoding as mlp_capability fields. This field is valid only when the <i>mlpn_status</i> indicates successful negotiation.	RO

28.19.29 PMLPN - Port Mellanox Link Protocol Negotiation

MLPN - Port Mellanox Link Protocol Negotiation

TODO: Add a section defining the MLPN flow.

The PMLPN register configures and reports the port Mellanox Link Protocol Negotiation mechanism.

If AN is disabled (PTYS), MLPN is also disabled regardless to PMLPN 'enable' bit.

Table 1058 - PMLPN - Port Mellanox Link Protocol Negotiation Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1059 - PMLPN - Port Mellanox Link Protocol Negotiation Register Fields

Offset	Bits	Name	Description	Access
00h	27:24	mlpn_status	Mellanox Link Properties Negotiation (MLPN) status. Indicates the status of the MLPN state machine: 0x0 - POLLING 0xF - DONE - Negotiation is done, link is configured accordingly.	RO
	23:16	local_port	Local port number.	Index
04h	31	e	Enable - enable MLPN mechanism. Enable value can only be updated while the port is disabled.	RW

28.19.30PVLC - Port Virtual Lane Capabilities

The PVLC register configures and reports the VLs capabilities.

Table 1060 - PVLC - Port Virtual Lane Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1061 - PVLC - Port Virtual Lane Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index
04h	3:0	vl_hw_cap	Virtual Lanes supported on this port. 0 - Reserved 1: VL0 2: VL0, VL1 [Internal] 3: VL0 - VL2 4: VL0 - VL3 [Internal] 5: VL0 - VL4 [Internal] 6: VL0 - VL5 [Internal] 7: VL0 - VL6 8: VL0 - VL7 15: VL0 - VL14	RO
08h	3:0	vl_admin	Virtual Lanes enabled by the local admin on this port. 0: No change; valid only on Set() 1: VL0 2: VL0, VL1 [Internal] 3: VL0 - VL2 4: VL0 - VL3 [Internal] 5: VL0 - VL4 [Internal] 6: VL0 - VL5 [Internal] 7: VL0 - VL6 8: VL0 - VL7 15: VL0 - VL14 Note: if changing vl_admin while the link is up , the vl_cap (IB spec) will be updated only after the link will go down and up.	RW

Table 1061 - PVLC - Port Virtual Lane Capabilities Register Fields

Offset	Bits	Name	Description	Access
0Ch	3:0	vl_operational	Operational VLs as configured by the SM on this port: 0: No change; valid only on Set() 1: VL0 2: VL0, VL1 [Internal] 3: VL0 - VL2 4: VL0 - VL3 [Internal] 5: VL0 - VL4 [Internal] 6: VL0 - VL5 [Internal] 7: VL0 - VL6 8: VL0 - VL7 15: VL0 - VL14 Changing vl_operational in certain PortStates may cause in flow control update errors which may initiate Link/PHY retraining. vl_operational value is valid only when link state is UP.	RO

28.19.31PPSC - Port Power Save Control

The PPSC register enables configuration of the link power save features including Dynamic speed and width.

Table 1062 - PPSC - Port Power Save Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
								local_port																									00h		
																															mod_pwr_opt	04h			
																																08h			
																																0Ch			
																														wrps_admin		10h			
																														wrps_status		14h			
	down_th_vld							up_threshold																		down_threshold									18h
	up_th_vld																															1Ch			
																														srps_admin		20h			
																														srps_status		24h			
																																28h			
																																2Ch			

Table 1063 - PPSC - Port Power Save Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
04	0	mod_pwr_opt	See <i>PCAM.feature_cap_mask bit</i> 16 for support indication: Transceiver power optimization: 0 - Disable 1 - Enable	RW
10h	3:0	wrps_admin	Width reduction power save admin state: 0000: Disable Width Reduction - Force link to full width (Default) 0001: Force Width Reduction - Force link to single lane 0010: Auto - Automatic Width Reduction (based on traffic).	RW

Table 1063 - PPSC - Port Power Save Control Register Fields

Offset	Bits	Name	Description	Access
14h	3:0	wrps_status	Link actual width: 0000: Full width 0001: Single lane 1111: In transition	RO
18h	31	up_th_vld	Valid bit to update the up threshold if '1' update <i>up_threshold</i>	WO
	30	down_th_vld	Valid bit to update the down threshold if '1' update <i>down_threshold</i>	WO
	23:16	up_threshold	Link width up threshold, Values: 1-5 (default 3) how aggressive/smoothed in response to high BW. 1 – means aggressive response to every BW increase 5 – means smoothed response to every BW increase	RW
	7:0	down_threshold	Link width down threshold, Values: 1%-100% The BW (In percent) before the link width is moved to single lane	RW
20h	3:0	srps_admin	Speed reduction power save admin state: 0000: Disable Speed Reduction - force link to full speed (Default) 0001: Force Speed Reduction - force link to low speed	RW
24h	3:0	srps_status	Link actual Speed: NOT SUPPORTED IN CURRENT FW	RO

28.19.32PLBF - Port Loopback Filter Register

f

Table 1064 - PLBF - Port Loopback Filter Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1065 - PLBF - Port Loopback Filter Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	2:0	lbf_mode	Loop-back filter mode: 0 - Do not update loop-back filter mode 1 - Disable loop-back filtering 2 - Enable loop-back filtering By default, loop-back filter is enabled for ALL ports	RW

28.19.33PIFR - Port Isolation Filter Register

Table 1066 - PIFR - Port Isolation Filter Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
table_id								local_port																								00h
																																04h-1Ch
port_filter																																20h-3Ch
port_filter_update_en																																40h-5Ch

Table 1067 - PIFR - Port Isolation Filter Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	table_id	[Internal] For Spectrum-3-phase2 Table ID: 0: Global table. Note that IGCR.pisbo affects the behavior of this table 1: BRIDGE_ONLY table	Index
00h	23:16	local_port	Local port number. Note: Port isolation is not supported to \ from port 0 (i.e CPU port).	Index
20h-3Ch	256	port_filter	Port filter bitmask 1 - Filter - Packets coming from <i>port_filter<i></i> will not be transmitted to port <i>local_port</i> . 0 - Do not filter (default value for all <i>local_port</i>). All LAG ports must share the same configuration. Refer to the relevant section for port numbering scheme.	RW
40h-5Ch	256	port_filter_update_en	port_filter_update_en bitmask, see port_filter When <i>port_filter_update_en<i></i> set, the filter value for local port i is updated according to <i>port_filter<i></i> . When <i>port_filter_update_en<i></i> is cleared, the filter value for local port i maintains its current value.	WO

28.19.34 PIPG - Port Inter Packet Gap

This register allows configuration of the port's inter packet gap insertion for Ethernet ports.

Table 1068 - PIPG - Port Inter Packet Gap Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
dic																																04h

Table 1069 - PIPG - Port Inter Packet Gap Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
04h	31	dic	Deficit Idle Count. When set, the dic bit is set, the Deficit Idle Count mechanism is enabled, else it is disabled. This bit is set by default.	RW
	5:2	ipg	Inter Packet Gap. Sets the Inter Packet Gap value for the ports, in 4 bytes granularity. Default value of this field is 3 (indicating 12 bytes IPG).	RW

28.19.35PPLR - Port Physical Loopback Register

This register allows configuration of the port's loopback mode.

In order to enable any loopback function, the port must be disabled first using PAOS register, configured to enable the Loopback using PPLR register and enabled using the PAOS register.

Note: This section is preliminary and subject to change. *This register allows configuration of the port's physical loopback mechanism.*

Table 1070 - PPLR - Port Physical Loopback Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1071 - PPLR - Port Physical Loopback Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0: Network_Port 1: Near-End_Port- (For Retimer/Gearbox - Host side) 2: Internal_IC_LR_Port 3: Far-End_Port - (For Retimer/Gearbox - Line side) 4: USR_Main 5: USR_Tile Other values are reserved.	Index
04h	23:16	lb_cap	Loopback capability - bitmask Bit 0: Phy_remote_loopback Bit 1: Phy_local_loopback - When set the port's egress traffic is looped back to the receiver and the port transmitter is disabled. Bit 2: External_local_loopback - Enables the port's transmitter to link with the port's receiver using an external loopback connector. Bit 3: sloop [Internal] Note: Phy remote loopback can be supported only for lane rate higher than 25Gbuaad.	RO
	7:0	lb_en	Loopback enable - bitmask. See lb_cap bits description.	RW

28.19.36PMPR - Port Module Properties Register

This register allows configuration of the port's module properties for backplane ports.

Table 1072 - PMPR - Port Module Properties Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1073 - PMPR - Port Module Properties Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
04h	7:0	attenuation_5g	Attenuation - total channel attenuation @ 5GHz in db.	RW
08h	7:0	attenuation_7g	Attenuation - total channel attenuation @ 7GHz in db. A 0 value implies that attenuation_5g is used also for 7GHz	RW
0Ch	7:0	attenuation_12g	Attenuation - total channel attenuation @ 12GHz in db. A 0 value implies that attenuation_5g is used also for 12GHz	RW

28.19.37 PEIR - Port Error Injection Register

This register allows configuration of the port's error injection mechanism.

This register is not supported on Switch-IB and above switches.

Table 1074 - PEIR - Port Error Injection Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1075 - PEIR - Port Error Injection Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index
04h	19:16	error_count	Error Count Number of times the port will generate the configured error. 0 indicates not to generate error. Reading <i>error_count</i> will return the number of left errors to inject. When 0, all requested errors has been injected.	RW
08h	19:16	lane	Bit vector for lane (bit 0, lane 0) Only relevant for specific errors at physical layer	RW
	7:0	error_type	Error Type to generate 0 - No Error ---- Physical Errors ---- 1 - 8/10 Symbol Error (per lane) 2 - 8/10 Disparity Error (per lane) 3:7 - Reserved 8 - 64/66 Sync Header (per lane, with block count) 9 - 64/66 Alignment Marker (per lane, with block count) 10 - 64/66 Block Type (per lane, with block count) [Internal] EDR only 12 - EDR FEC correctable symbol error 13 - EDR FEC un-correctable symbol error ---- Packet Errors ---- 16 - EBP marking (with packet count) 17 - Corrupt data packet ICRC 18 - Corrupt data packet VCRC 19 - Corrupt credit packet LPCRC [Internal] We can implement the below, but control should be different. Maybe for pre-defined time or set and clear. Stop sending Comma/Skip symbols RX Discard per VL / Buffer TX Discard per VL / TClass Transmit Hold per VL / TClass	RW

28.19.38PTER - Port Transmit Errors Register

This register allows configuration of the port's error injection mechanism.

Table 1076 - PTER- Port Transmit Errors Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
				error_page				local_port								pnat								port_type				status			00h						
page_data																																					04h-1Ch

Table 1077 - PTER- Port Transmit Errors Register Fields

Offset	Bits	Name	Description	Access
00h	27:24	error_page	Error_page selection: 0 - Injection of Phy Errors 1 - Injection of Port Errors	Index
	23:16	local_port	Local port number	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Reserved 2 - Internal IC LR Port 3 - Reserved 4 - USR Main 5 - USR Tile Other values are reserved.	Index
	3:0	status	Error injection status: 0x0 -Good status 0x2 - No available sources for Error injection 0x3 - Error injection configuration when port is in non operational state (port is neither in UP or TEST MODE) 0x4 - out of range BER setting	RO
04h-1Ch	224	page_data	error injection page data: See Table 1078, "PTER- Port Transmit Errors Register Phy Level Layout," on page 1271 See Table 1080, "PTER- Port Transmit Errors Register Port Level Fields," on page 1272	

Table 1078 - PTER- Port Transmit Errors Register Phy Level Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h-18h

Table 1079 - PTER- Port Transmit Errors Register Phy Level Layout

Offset	Bits	Name	Description	Access
00h	27:24	error_type_oper	Error injection operational status 0x0 - No error injection 0x1 - Performing error injection	RO
	15:8	error_type_cap	Error injection capabilities Bit 0 - Raw BER Bit 1 - Effective BER [Internal] Bit 2 - RS-FEC symbol error Bit 3 - RS-FEC uncorrectable codewords error	RO
	7:0	error_type_admin	Error Type to generate 0 - No Error ---- Physical Errors ---- 1 - Raw BER 2 - Effective BER 4- RS-FEC symbol error 8- RS-FEC uncorrectable codewords error	RW

Table 1079 - PTER- Port Transmit Errors Register Phy Level Layout

Offset	Bits	Name	Description	Access
04h	27:12	error_injection_time	Duration in 10 msec the port will generate errors. Reading this field will return the time left for errors to inject in msec [Internal] For <i>error_type = 4</i> OR <i>error_type = 8 (RS-FEC symbols)</i> <i>error_injection_time</i> represent the number of times to produce the <i>rs_errors_interval</i> . Reading this field will return the number of <i>rs_errors_interval</i> rounds left for errors to inject. 0x0000 indicates no generation of errors. When set to 0xFFFF - No decremental operation, meaning the errors will be injected continuously without stop condition.	RW
	11:8	ber_mantissa	BER for injection. <i>BER = ber_mantissa * 10^(-ber_exp)</i>	RW
	7:0	ber_exp		RW
08h	15:0	rs_symbol_errors	Valid when <i>error_type_admin = 4,8</i> When <i>error_type_admin = 4,8</i> Number of times the port will generate the configured error RS-FEC errors. 0 indicates not to generate error.	RW
0Ch	15:0	rs_errors_interval	Valid when <i>error_type_admin = 4,8</i> When <i>error_type_admin = 4,8</i> Number of codewords which will be injected with <i>rs_symbol_errors</i> errors.	RW

Table 1080 - PTER- Port Transmit Errors Register Port Level Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h-18h

Table 1081 - PTER- Port Transmit Errors Register Port Level Errors Fields

Offset	Bits	Name	Description	Access
00h	15:8	error_type_cap	Error injection capabilities Bit 0- Corrupt data packet ICRC Bit 1- Corrupt data packet VCRC Bit 2- Corrupt credit packet LPCRC	RO
	7:0	error_type_admin	Error Type to generate 0 - No Error ---- Port Errors ---- 1 - Corrupt data packet ICRC 2 - Corrupt data packet VCRC 4- Corrupt credit packet LPCRC	RW
04h	16:12	error_count	Error Count Number of times the port will generate the configured error. 0 indicates not to generate error. Reading error_count will return the number of left errors to inject. When 0, all requested errors has been injected.	RW

28.19.39 PLPC - Phy Link Profiles Control

The PLPC register configures and reports the link mode profiles - which controls the FEC mode and retransmission mode. Each *profile_id* is associated with a port type, for ports that enable media based profile, the *profile_id* is assigned based on the port type. The association of a port to a profile is done using the

register. The register can be accessed multiple times in order to control different properties for different speeds or link protocols. In addition, a configuration can be applied concurrently to multiple profiles, link protocols and speeds by setting multiple profiles, link protocols and speeds in the same register access. Note that changes to the link extended capabilities while the link is up take effect only after link retrain.

The selection of the FEC profile to be based on IB spec or media dependent is done by setting bits 1 and 0 of the “*port_profile_mode*” in *PPLM* register (auto).

Requesting override FEC in IB is done by clearing bits 1 and 0 of the “*port_profile_mode*” in *PPLM* register and setting bit 7 or bit 8 of “*profile_id*” in *PLPC*.

Bit 7 - NO-FEC; Bit 8 - FEC-ON.

Table 1082 - PLPC - Phy Link Profiles Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				profile_id																proto_mask												00h
																lane_speed																04h
																																08h
retransmission_capability								fec_mode_capability																								0Ch
retransmission_support_admin								fec_mode_support_admin																								10h
retransmission_request_admin								fec_mode_request_admin																								14h
																																18h-24h

Table 1083 - PLPC - Phy Link Profiles Control Register Fields

Offset	Bits	Name	Description	Access
00h	27:16	profile_id	Profile ID (bitmask) Bit 0 - IB spec / legacy (See profiles description) Bit 1 - Internal ports (Backplane) Bit 2 - Passive copper - Short Bit 3 - Passive copper - Medium Bit 4 - Passive copper - Long Bit 5 - Active optics / copper short reach (<30m) Bit 6 - Optics long reach (>30m) Bit 7 - NO-FEC Bit 8 - FEC-ON Bit 9 - User defined 1 (not supported) Bit 10-11 - Reserved Note: setting multiple bits will apply the setting to multiple profile ids. Note: Short, Medium, Long are functions of the cable length and attenuation.	Index
	11:8	proto_mask	Protocol Mask. Indicates which of the protocol data is valid Bit 0: InfiniBand Bit 1: Reserved Bit 2: Ethernet Note: setting multiple bits will apply the setting to all the set protocols	Index
04h	15:0	lane_speed	Per lane speed (bitmask) Bit 0 - Reserved Bit 1 - Reserved Bit 2 - Reserved Bit 3 - FDR10 (10GE / 40GE) Bit 4 - FDR (56GE) Bit 5 - EDR (25GE / 50GE / 100GE) Note: setting multiple bits will apply the setting to all the set speeds	Index
08h	8	lpbf	Link Performance Based FEC When set, only FEC types that are applicable for the link performance will be used. When cleared, the link performance is ignored for the FEC selection and the FEC is selected only according to the policy. If the selected FEC is not applicable for the link performance, the speed may be considered as non-operational.	RW
	7:0	fec_mode_policy	FEC decision policy: 0 - optimize robustness 1 - optimize latency 2 - optimize bandwidth	RW

Table 1083 - PLPC - Phy Link Profiles Control Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:24	retransmission_capability	Retransmission capability (bitmask) Bit 0 - LLR Bit 1 - PLR	RO
	23:0	fec_mode_capability	FEC capability (bitmask) Bit 0 - No FEC Bit 1 - Firecode FEC Bit 2 - Standard RS-FEC - RS(528,514) Bit 3 - Standard LL RS-FEC - RS(271,257) Bit 4 - Mellanox Strong RS-FEC - RS(277,257) Bit 5 - Mellanox LL RS-FEC - RS(163,155) Bit 6 - Correct on Demand (COD)	RO
10h	31:24	retransmission_support_admin	Retransmission support admin (bitmask) When cleared peer ports requests to enable the retransmission mode will be ignored. Uses the same encoding as retransmission_capability.	RW
	23:0	fec_mode_support_admin	FEC support admin (bitmask) Valid only for Mellanox Phy only, when cleared peer ports requests to enable the FEC mode will be ignored. Uses the same encoding as fec_mode_capability. Writing 0 to the filed keeps the current setting.	RW
14h	31:24	retransmission_request_admin	Retransmission request admin (bitmask) Sets the retransmission modes which are administratively enabled. Uses the same encoding as retransmission_capability. The retransmission_request_admin must be a subset of retransmission_support_admin	RW
	23:0	fec_mode_request_admin	FEC request admin (bitmask) Sets the FEC modes which are administratively enabled. Uses the same encoding as retransmission_capability. Writing 0 to the filed keeps the current setting. The fec_mode_request_admin must be a subset of fec_mode_support_admin	RW

28.19.40PPLM - Port Phy Link Mode

The PPLM register configures and reports the link mode profile associated with the port and the ports active profile, FEC and retransmission modes.

NOTE: Setting the fec admin fields of this register shall only take effect during link training and negotiation. When set while the link is up, the changes will not take effect unless the port goes back to training and negotiation state (during transition from down to up state).

Software can force training by disabling and enabling the port..

Table 1084 - PPLM - Port Phy Link Mode Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
								local_port												port_type															00h
																																	04h		
port_profile_mode								static_port_profile								active_port_profile								fec_policy_ cap				fec_policy				08h			
retransmission_active								fec_mode_active																								0Ch			
rs_fec_correc- tion_bypass_ cap													fec_over- ride_cap_56g				fec_over- ride_ cap_100g				fec_over- ride_cap_50g				fec_override_ cap_25g				fec_over- ride_ cap_10g_40g				10h		
rs_fec_correc- tion_by- pass_admin													fec_over- ride_admin- _56g				fec_over- ride_admin- _100g				fec_over- ride_admin- _50g				fec_over- ride_admin- _25g				fec_over- ride_admin- _10g_40g				14h		
fec_override_cap_400g_8x																fec_override_cap_200g_4x																18h			
fec_override_cap_100g_2x																fec_override_cap_50g_1x																1Ch			
fec_override_admin_400g_8x																fec_override_admin_200g_4x																20h			
fec_override_admin_100g_2x																fec_override_admin_50g_1x																24h			
																																28h-2Ch			
ib_fec_override_cap_hdr																ib_fec_override_cap_edr																30h			
ib_fec_override_cap_fdr																ib_fec_override_cap_fdr10																34h			
ib_fec_override_admin_hdr																ib_fec_override_admin_edr																38h			
ib_fec_override_admin_fdr																ib_fec_override_admin_fdr10																3Ch			
																																40h-4Ch			

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index
	11:8	port_type	Supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Gearbox - Host side) 2 - Internal IC Port 3 - Far-End Port (For Gearbox - Line side) Other values are reserved.	Index
04h	31:0	reserved		
08h	31:24	port_profile_mode	Port profile modes (bitmask): Bit 0: Port_type_based_port_profile - When set the profile will be selected according to the connected port type. When cleared the port's profile will be set according to the static_port_profile field Bit 1: Legacy_or_IB_spec_profile_enable - (default set) when set, the IB spec / legacy profile will be applied on the port when connected to a peer non-Mellanox or legacy port. Note that when Legacy / IB spec profile enable bit is cleared, some of the configurations may not be supported by the peer port and will be ignored in the negotiation. Bit 2: Ethernet_Enhanced_FEC - (default cleared) TBD exact text - When auto mode (no override) is set on Mellanox links select the FEC based on the profile including Mellanox Phy modes Bits 7-3 - Reserved	RW
	23:16	static_port_profile	Valid when Media based port profile is cleared, used to set the static port's profile index.	RW
	15:8	active_port_profile	The port's active Profile ID. 0xFF - unassigned - set when Media based port profile is set and no medium is connected.	RO
	7:4	fec_policy_cap	FEC policy capability (inverted logic: 0 - supported, 1 - not supported): Bit 0: Latency_Policy Bit 1: Robust_without_retransmission_Policy Bit 2: Robust_with_retransmission_Policy Bit 3: Standard_Policy	RO
	3:0	fec_policy	FEC policy (see <i>fec_policy_cap</i> for supported policy allowed configurations): 0: Latency_Policy 1: Robust_without_retransmission_Policy 2: Robust_with_retransmission_Policy 3: Standard_Policy Note: It is recommended to set all ports with the same <i>fec_policy</i> value.	RW
0Ch	31:24	retransmission_active	Active Retransmission (bitmask) Bit 0: LLR Bit 1: Reserved - (PLR)	RO

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
0Ch	23:0	fec_mode_active	Active FEC (bitmask) 0: No_FEC 1: Firecode_FEC 2: Standard_RS_FEC - RS(528,514) 3: Standard_LL_RS-FEC - RS(271,257) 4: Mellanox_Strong_RS-FEC - RS(277,257) [Internal] 5: Mellanox_LL_RS-FEC - RS(163,155) [Internal] 6: Reserved - [Internal] 7: Standard_RS-FEC - (544,514) 8: Zero_Latency_FEC - [Internal] 9: Ethernet_Consortium_LL_50G_RS-FEC - (272,257+1) 12: RS-FEC - (544,514) + PLR [Internal] 13: LL-FEC - (271,257) + PLR [Internal]	RO
10h	31:28	rs_fec_correction_bypass_cap	RS-FEC correction bypass override capability: 0: NO_correction_bypass 1: RS-FEC_correction_bypass	RO
	19:16	fec_override_cap_56g	56GE Ethernet FEC override capability bitmask: 0: No-FEC 1: Firecode_FEC 2: Reserved2 3: Reserved3	RO
	15:12	fec_override_cap_100g	100GE (4 lanes) Ethernet FEC override capability bitmask: 0: No-FEC 1: Reserved1 2: RS-FEC - (528,514) 3: Reserved3	RO
	11:8	fec_override_cap_50g	50GE (2 lanes) Ethernet FEC override capability bitmask: 0: No-FEC 1: Firecode_FEC 2: RS-FEC 3: Reserved	RO
	7:4	fec_override_cap_25g	25GE Ethernet FEC override capability bitmask: 0: No-FEC 1: Firecode_FEC 2: RS-FEC 3: Reserved	RO
	3:0	fec_override_cap_10g_40g	10GE/40GE Ethernet FEC override capability bitmask: 0: No-FEC 1: Firecode_FEC 2: Reserved2 3: Reserved3	RO

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
14h	31:28	rs_fec_correction_bypass_admin	RS-FEC correction bypass override admin: (one-hot setting): 0: auto_mode - no override 1: NO_correction_bypass 2: RS-FEC_correction_bypass	RW
	19:16	fec_override_admin_56g	56GE Ethernet FEC override admin, see 56g_fec_override_cap. (one-hot setting): 0: auto_mode - no override 1: No_FEC 2: Firecode_FEC	RW
	15:12	fec_override_admin_100g	100GE Ethernet FEC override admin, see 100g_fec_override_cap. (one-hot setting): 0: auto_mode - no override 1: No_FEC 4: RS-FEC	RW
	11:8	fec_override_admin_50g	50GE Ethernet FEC override admin, see 50g_fec_override_cap. (one-hot setting): 0: auto_mode - no override 1: No_FEC 2: Firecode_FEC 4: RS-FEC Note - 25g_fec_override_admin and 50g_fec_override_admin must be set with the same value.	RW
	7:4	fec_override_admin_25g	25GE Ethernet FEC override admin, see 25g_fec_override_cap. (one-hot setting): 0: auto_mode - no override 1: No_FEC 2: Firecode_FEC 4: RS-FEC Note - 25g_fec_override_admin and 50g_fec_override_admin must be set with the same value.	RW
	3:0	fec_override_admin_10g_40g	10GE/40GE Ethernet FEC override admin, see 10g_40g_fec_override_cap. (one-hot setting): 0: auto_mode - no override 1: No_FEC 2: Firecode_FEC	RW

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
18h	31:16	fec_override_ cap_400g_8x	400GE FEC override capability bitmask: Bit 0 - Reserved Bit 1 - Reserved Bit 2 - Reserved Bit 3 - Reserved // [Internal] LL-FEC (271,257) Bit 4 - Reserved Bit 5 - Reserved Bit 6 - Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC Bit 9- 50G Ethernet Consortium LL RS-FEC (272,257+1) [Internal] Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR [Internal] Bit 14 - ELL (272,257+1) + PLR	RO
	15:0	fec_override_ cap_200g_4x	200GE FEC override capability bitmask: Bit 0 - Reserved Bit 1 - Reserved Bit 2 - Reserved Bit 3 - Reserved // [Internal] LL-FEC (271,257) Bit 4 - Reserved Bit 5 - Reserved Bit 6 - Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC Bit 9- 50G Ethernet Consortium LL RS-FEC (272,257+1) [Internal] Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR [Internal] Bit 14 - ELL (272,257+1) + PLR	RO

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
1Ch	31:16	fec_override_cap_100g_2x	100GE over 2 lanes, FEC override capability bitmask: Bit 0 - No-FEC Bit 1- Reserved Bit 2- Reserved Bit 3- Reserved // [Internal] LL-FEC (271,257) Bit 4- Reserved Bit 5- Reserved Bit 6- Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC Bit 9- 50G Ethernet Consortium Low Latency RS-FEC (272,257+1) [Internal] Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR [Internal] Bit 14 - ELL (272,257+1) + PLR	RO
	15:0	fec_override_cap_50g_1x	50GE single lane override capability bitmask: Bit 0 - Reserved Bit 1- Reserved Bit 2- Reserved Bit 3- Reserved // [Internal] LL-FEC (271,257) Bit 4- Reserved Bit 5- Reserved Bit 6- Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC Bit 9- 50G Ethernet Consortium LL RS-FEC (272,257+1) [Internal] Bit 12 - RS-FEC - (544,514) + PLR [Internal] Bit 13 - LL-FEC - (271,257) + PLR [Internal] Bit 14 - ELL - (272,257+1) + PLR	RO
20h	31:16	fec_override_admin_400g_8x	400GE FEC override admin bitmast (one-hot setting): 0: Auto_mode 7: RS-FEC_(544,514) 8: Zero_Latency_FEC 9: RS-FEC_(272,257+1) 12: RS-FEC_(544,514) - + PLR [Internal] 13: LL-FEC_(271,257) - + PLR [Internal] 14: LL-FEC_(272,257+1) - + PLR [Internal]	RW
	15:0	fec_override_admin_200g_4x	200GE FEC override admin bitmask (one-hot setting) 0: Auto_mode 7: RS-FEC - (544,514) [Internal] Bit 8 is set - Zero Latency FEC Bit 9 is set - RS-FEC (272,257+1) [Internal] Bit 12 is set - RS-FEC (544,514) + PLR [Internal] Bit 13 is set - LL-FEC (271,257) + PLR [Internal] Bit 14 is set - LL-FEC (272,257+1) + PLR	RW

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
24h	31:16	fec_override_admin_100g_2x	100GE, 2 lanes FEC override admin bitmask (one-hot setting): 0: Auto_mode Bit 0 is set - Reserved Bit 2 is set - Reserved Bit 7 is set- RS-FEC (544,514) [Internal] Bit 8 is set - Zero Latency FEC Bit 9 is set - RS-FEC (272,257+1) [Internal] Bit 12 is set - RS-FEC (544,514) + PLR [Internal] Bit 13 is set - LL-FEC (271,257) + PLR [Internal] Bit 14 is set - LL-FEC (272,257+1) + PLR	RW
	15:0	fec_override_admin_50g_1x	50GE, 1lanes FEC override admin bitmask (one-hot setting) 0 - Auto mode Bit 0 is set - NO-FEC Bit 1 is set - FC-FEC Bit 2 is set - RS-FEC (528,514) Bit 7 is set- RS-FEC (544,514) [Internal] Bit 8 is set - Zero Latency FEC Bit 9 is set - RS-FEC (272,257+1) [Internal] Bit 12 is set - RS-FEC (544,514) + PLR [Internal] Bit 13 is set - LL-FEC (271,257) + PLR [Internal] Bit 14 is set - LL-FEC (272,257+1) + PLR	RW
30h	31:16	ib_fec_override_cap_hdr	HDR FEC override capability bitmask: Bit 0 - No-FEC Bit 1- Reserved Bit 2- Reserved Bit 3- LL-FEC (271,257) Bit 4- Reserved Bit 5- Reserved Bit 6- Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC ~~[Internal]-~~ Bit 12 - RS-FEC (544,514) + PLR ~~[Internal]-~~ Bit 13 - LL-FEC (271,257) + PLR	RO
	15:0	ib_fec_override_cap_edr	EDR FEC override capability bitmask: Bit 0 - No-FEC Bit 1- Reserved Bit 2- RS-FEC (528,514) Bit 3- LL-FEC (271,257) Bit 4- Reserved Bit 5- Reserved Bit 6- Reserved Bit 7 - RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC [Internal]-Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR	RO

Table 1085 - PPLM - Port Phy Link Mode Register Field

Offset	Bits	Name	Description	Access
34h	31:16	ib_fec_override_cap_fdr	FDR FEC override capability bitmask: Bit 0: No_FEC Bit 1: Firecode_FEC Bit 2-15: Reserved	RO
	15:0	ib_fec_override_cap_fdr10	FDR FEC10 override capability bitmask: Bit 0: No_FEC Bit 1: Firecode_FEC Bit 2-15: Reserved	RO
38h	31:16	ib_fec_override_admin_hdr	HDR FEC override admin bitmask: 0 - Auto mode Bit 0 is set - No FEC Bit 4 is set- LL-FEC (271,257) Bit 7 is set- RS-FEC (544,514) [Internal] Bit 8 is set - Zero Latency FEC ~~[Internal]~~ Bit 12 is set - RS-FEC (544,514) + PLR ~~[Internal]~~ Bit 13 is set - LL-FEC (271,257) + PLR	RW
	15:0	ib_fec_override_admin_edr	EDR FEC override admin bitmask: 0 - Auto mode Bit 0 is set - No FEC Bit 4 is set- LL-FEC (271,257) Bit 7 is set- RS-FEC (544,514) [Internal] Bit 8 is set - Zero Latency FEC [Internal] Bit 12 is set - RS-FEC (544,514) + PLR [Internal] Bit 13 is set - LL-FEC (271,257) + PLR	RW
3Ch	31:16	ib_fec_override_admin_fdr	FDR FEC override admin bitmask: 0: Auto_mode 1: No_FEC 2: Firecode_FEC	RW
	15:0	ib_fec_override_admin_fdr10	FDR FEC override admin bitmask: 0:Auto_mode 1: No_FEC 2: Firecode_FEC	RW

28.19.41 PCNR- Phy CoNfiguration Register

The PCNR register configures per phy characteristics.

Table 1086 - PCNR- Phy CoNfiguration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port																							tuning_override	00h
																															keep_phy_setting	04h
																																08h

Table 1087 - PCNR- Phy CoNfiguration Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	0	tuning_override ¹	When Set, port will override tuning process upon following link-up command (PAOS.admin_status = UP) Cleared by FW once PAOS.admin_status = UP command is set. The register can be set only when port operational status is UP (PAOS.oper_status=0x1) NOTE: if physical environment was changed (i.e. replacement of module, temp change, etc) there is a possibility that link won't be established or will be established with high BER	WO
04	0	keep_phy_setting	When Set, The port will keep the same phy setting upon link down event that occurs only upon link down command of <u>peer port</u>. In the event of Down command/cable disconnect, entire link up flow will be initialized. NOTE: This mode can be configured only when PTYS.an_disable_admin is set (i.e. AN is disabled). NOTE: if physical environment was changed (i.e. replacement of module, temp change, etc) there is a possibility that link won't be established or will be established with high BER	WO

1. [Internal] The tuning override is not return to 0 upon sw_reset

[internal] PPAMP - Port Phy opAMP data

The PPAMP register is used in order to retrieve the opamp data. The opamp data is accessed on a per port (4X) RX/TX data.

Table 1088 - [Internal] PPAMP - Port Phy opAMP Data Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								opamp_group																opam- p_group_ - type				00h				
start_index																				num_of_indices												04h
								max_opamp_group												max_num_of_indices												08h
index_data[0]																index_data[1]												0Ch				
index_data[2]																index_data[3]												10h				
...																																..
index_data[14]																index_data[15]																28h

Table 1089 - [internal] PAMP - Port Phy opAMP Data Register Field

Offset	Bits	Name	Description	Access
00h	23:16	opamp_group	See opamp_group_type. opamp_group valid values are 0 - max_opamp_group	Index
	3:0	opamp_group_type	Determines the opamp group type which opamp_group points to: 0 - Network ports OpAMPs. When selected the network ports' OpAMPs are accessed. The opamp_group holds an index to the network port's 4X Serdes. 1 - PCI OpAMPs When selected the PCI's OpAMPs are accessed. The opamp_group holds an index to the PCI 4X Serdes. 2 - PLL OpAMPs When selected the PLL's OpAMPs are accessed. The opamp_group indicates the PLL index that is accessed.	Index
04h	31:16	start_index	Start index for the block of opamp data	RW
	15:12	reserved	reserved for future command field	
	11:0	num_of_indices	number of opamp indices to access	RW

Table 1089 - [internal] PAMP - Port Phy opAMP Data Register Field

Offset	Bits	Name	Description	Access
08h	23:16	max_opamp_group	The maximum opamp_group index valid for the opamp_group_type. A value of 0xFF indicates that the opamp_group_type is not supported	RW
	11:0	max_num_of_indices	The maximum OpAmp index valid for the opamp_group	RW
0Ch-28h	31:16	index_data[16]	Opamp index i data. Even indices are in offset 31:16, odd indices are in offsets 15:0.	RW

28.19.42[Internal] SLRP - Serdes Lane Receive Parameters

This register enables to read or set the serdes receive parameters.

When read, the SLRP returns the receive setting for the lane. When the link is up or down the last high speed setting is returned.

Table 1090 - SLRP - Serdes Lane Receive Parameters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status				version				local_port				pnat		gb		lane				port_type								00h				
page_data																																04h-24h

Table 1091 - SLRP - Serdes Lane Receive Parameters Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 40nm products 1 - 28nm products 3 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	13	gb	[Internal]: Set to '1' for Gearbox control and access. When set, see the relevant <i>page_data</i> .	OP
	11:8	lane	Physical lane number [Internal]: When <i>gb</i> set to '1', bits 10:8 of <i>lane</i> number represent the channel number of the Gearbox. Bit 11 chooses between Host ('1') / Line side ('0').	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index

Table 1091 - SLRP - Serdes Lane Receive Parameters Register Fields

Offset	Bits	Name	Description	Access
04h-24h	288	page_data	See Table 1092, “[Internal] Serdes Lane Receive Parameters Register Layout for 40nm and 28nm,” on page 1289 See Table 1094, “[Internal] Serdes Lane Receive Parameters Register Layout for 16nm,” on page 1291 See Table 1096, “[Internal] Serdes Lane Receive Parameters Register Layout For Gearbox,” on page 1295 See Table 1098, “[Internal] SLRP - Serdes Lane Receive Parameters Register Layout for USR,” on page 1296	RO

Table 1092 - [Internal] Serdes Lane Receive Parameters Register Layout for 40nm and 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ib_sel																			dp_sel	dp90sel			mix90phase							00h		
ffe_tap0								ffe_tap1								ffe_tap2								ffe_tap3							04h	
ffe_tap4								ffe_tap5								ffe_tap6								ffe_tap7							08h	
ffe_tap8								mixerbias_tap_amp														ffe_tap_en							0Ch			
ffe_tap_offset0								ffe_tap_offset1								slicer_offset0												10h				
mixer_offset0																mixer_offset1												14h				
mixerbgn_inp								mixerbgn_inn								mixerbgn_refp								mixerbgn_refn							18h	
sel_slicer_lctrl_h	sel_slicer_lctrl_l		ref_mixer_vreg					slicer_gctrl								lctrl_input								mixer_offset_cm1							1Ch	
common_mode						mixer_offset_cm0														slicer_offset_cm							20h					

Table 1093 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for 40nm and 28nm

Offset	Bits	Name	Description	Access
00	31:30	ib_sel	input buffer select: 0 - Active 1 - Inactive 2 - IB0 3 - IB1	Index
	12	dp_sel	datapath select, indicates the input buffer connected to the data path.	RW
	11:8	dp90sel		RW
	7:0	mix90phase		RW
04h	31:24	ffe_tap0		RW
	23:16	ffe_tap1		
	15:8	ffe_tap2		
	7:0	ffe_tap3		
08h	31:24	ffe_tap4		
	23:16	ffe_tap5		
	15:8	ffe_tap6		
	7:0	ffe_tap7		
0Ch	31:24	ffe_tap8		RW
	23:16	mixer-bias_tap_amp		RW
	8:0	ffe_tap_en		RW
10h	31:24	ffe_tap_offset0		RW
	23:16	ffe_tap_offset1		RW
	15:0	slicer_offset0		RW
14h	31:16	mixer_offset0		RW
	15:0	mixer_offset1		RW
18h	31:24	mixerbgn_inp		RW
	23:16	mixerbgn_inn		RW
	15:8	mixerbgn_refp		RW
	7:0	mixerbgn_refn		RW

28nm

Table 1093 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for 40nm and

Offset	Bits	Name	Description	Access
1Ch	31	sel_slicer_lctrl_h		RW
	30	sel_slicer_lctrl_l		RW
	28:24	ref_mixer_vreg		RW
	23:16	slicer_gctrl		RW
	15:8	lctrl_input		RW
	7:0	mixer_offset_cm1		RW
20h	31:26	common_mode		RW
	24:16	mixer_offset_cm0		RW
	8:0	slicer_offset_cm		RW

Table 1094 - [Internal] Serdes Lane Receive Parameters Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
mixer_offset_up								mixer_offset_down								sel_enc														dp_sel	00h	
sel_ref_in_mixer_biasgen_up				sel_ref_in_mixer_biasgen_mid				sel_ref_in_mixer_biasgen_down				sel_ref_in_mixer_biasgen_eo				sel_ref_out_mixer_biasgen_up				sel_ref_out_mixer_biasgen_mid				sel_ref_out_mixer_biasgen_down				sel_ref_out_mixer_biasgen_eo				04h
sel_amp_mixer_biasgen_up	sel_amp_mixer_biasgen_mid	sel_amp_mixer_biasgen_down	sel_amp_mixer_biasgen_eo	mixer_offset_eye_scan								gctrl_bin_bgn0_n				gctrl_bin_bgn0_p				gctrl_bin_bgn1_n				gctrl_bin_bgn1_p				08h				
sel_digital_gctrl	sel_bias2ctlc							selg_c_ref_op0_alev_ctlc				selg_c_ref_op1_alev_ctlc				sel_bias_en_0				sel_bias_en_1				sel_bias_en_2					0Ch			
sel_bias_en_3				sel_bias_en_4				sel_bias_en_5				sel_bias_en_6				sel_gctrl_n_en_0				sel_gctrl_n_en_1				sel_gctrl_n_en_2					10h			
sel_gctrl_n_en_3				sel_gctrl_n_en_4				sel_gctrl_n_en_5				sel_gctrl_n_en_6				sel_gctrl_n_en_7				sel_gctrl_n_en_8												
sel_gctrl_n_en_3				sel_gctrl_n_en_4				sel_gctrl_n_en_5				sel_gctrl_n_en_6				sel_gctrl_n_en_7				sel_gctrl_n_en_8				sel_gctrl_n_en_9				14h				

Table 1094 - [Internal] Serdes Lane Receive Parameters Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sel_gc-trlp_en_4				sel_gc-trlp_en_5				sel_gc-trlp_en_6				crnt_bgn_off-set_p1				crnt_bgn_off-set_n1				crnt_bgn_off-set_p0				crnt_bgn_off-set_n0				18h				

Table 1095 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for 16nm

Offset	Bits	Name	Description	Access
00h	31:24	mixer_offset_up		RW
	23:16	mixer_offset_down		RW
	15:8	sel_enc		RW
	1:0	dp_sel	Datapath select, indicates the input buffer that will not be active in the data path. 0 - Up 1 - Mid 2 - Down 3 - None Note: for a change, need to return to default settings, and only after that the new selection will take a place. Note: PCIe uses NRZ thus, only the middle input buffer is used. All options of 0-2 should trigger the same result (same as '1' - Mid).	RW
The fields in offset 08h - 24h hold the selected input buffer info				

Table 1095 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for 16nm

Offset	Bits	Name	Description	Access
04h	31:28	sel_ref_in_mixer_biasgen_up		RW
	27:24	sel_ref_in_mixer_biasgen_mid		RW
	23:20	sel_ref_in_mixer_biasgen_down		RW
	19:16	sel_ref_in_mixer_biasgen_eo		RW
	15:12	sel_ref_out_mixer_biasgen_up		RW
	11:8	sel_ref_out_mixer_biasgen_mid		RW
	7:4	sel_ref_out_mixer_biasgen_down		RW
	3:0	sel_ref_out_mixer_biasgen_eo		RW
08h	31:30	sel_amp_mixer_biasgen_up		RW
	29:28	sel_amp_mixer_biasgen_mid		RW
	27:26	sel_amp_mixer_biasgen_down		RW
	25:24	sel_amp_mixer_biasgen_eo		RW
	23:16	mixer_offset_eye_scan		RW
	15:12	gctrl_bin_bg0_n	Number of parallel resistors for peaking control (replica), p/n same values.	RW
	11:8	gctrl_bin_bg0_p		RW
	7:4	gctrl_bin_bg1_n		RW
	3:0	gctrl_bin_bg1_p		RW

Table 1095 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for 16nm

Offset	Bits	Name	Description	Access
0Ch	31	sel_digital_gctrl	0 - Analog control 1 - Digital control	RW
	30:24	sel_bias2ctle	Selects which group goes to picking control. Bit per stage.	RW
	23:19	selg- c_ref_op0_alev- ctle	Determines peaking, higher value means less peaking.	RW
	18:14	selg- c_ref_op1_alev- ctle		RW
	13:10	sel_bias_en_0	CTLE stage gain. Higher value means higher gain. 1-16 stages in parallel.	RW
	9:6	sel_bias_en_1		RW
	5:2	sel_bias_en_2		RW
10h	31:28	sel_bias_en_3		RW
	27:24	sel_bias_en_4		RW
	23:20	sel_bias_en_5		RW
	19:15	sel_bias_en_6		RW
	14:11	sel_gctrln_en_0		RW
	10:7	sel_gctrln_en_1		RW
	6:3	sel_gctrln_en_2		RW
14h	31:28	sel_gctrln_en_3		RW
	27:24	sel_gctrln_en_4		RW
	23:20	sel_gctrln_en_5		RW
	19:16	sel_gctrln_en_6		RW
	15:12	sel_gctrlp_en_0		RW
	11:8	sel_gctrlp_en_1		RW
	7:4	sel_gctrlp_en_2		RW
	3:0	sel_gctrlp_en_3		RW
18h	31:28	sel_gctrlp_en_4		RW
	27:24	sel_gctrlp_en_5		RW
	23:20	sel_gctrlp_en_6		RW
	19:15	crnt_bgn_off- set_p1	Offset of biasgen_p1	RO

Table 1096 - [Internal] Serdes Lane Receive Parameters Register Layout For Gearbox

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																				ffe_tap_index						ffe_sub_channel							00h
snr_en					dsp_mode							afe_trim									ctle							04h					
raw_snr_value																				dsp_err_gen					snr_sym_cnt		08h						
db_snr_value																																0Ch	
																								ffe_tap_value								10h	
																																14h-1Ch	

Table 1097 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields For Gearbox

Offset	Bits	Name	Description	Access
00h	11:8	ffe_tap_index	0: Pre-cursor 3 tap index 1: Pre-cursor 2 tap index 2: Pre-cursor 1 tap index 3: Main cursor tap index 4: Post-cursor 1 tap index 5: Post-cursor 2 tap index 6: Post-cursor 3 tap index 7: Post-cursor 4 tap index 8: Post-cursor 5 tap index 9: Post-cursor 6 tap index	Index
	4:0	ffe_sub_channel	One of the 32 FFE sub-channels	Index
04h	31	snr_en	SNR monitor enable.	RW
	26:23	dsp_mode		RW
	19:16	afe_trim		RW
	7:0	ctle		RW

Table 1097 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields For Gearbox

Offset	Bits	Name	Description	Access
08h	31:16	raw_snr_value	Raw SNR Value	RO
	9:8	dsp_err_gen	The point where the error signal is generated from in the datapath 1: Prior to the reflection cancellor block 2: After the reflection cancellor block 3: Default	WO
	2:0	snr_sym_cnt	SNR symbol count 0: 2^10 blocks 1: 2^13 blocks 2: 2^16 blocks 3: 2^19 blocks 4: Default	WO
0Ch	31:0	db_snr_value	SNR Value in mili-dB units	RO
10h	7:0	ffe_tap_value		RO

Table 1098 - [Internal] SLRP - Serdes Lane Receive Parameters Register Layout for USR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
mixer00						mixer90						termination																		00h						
neg_- cap_u p	neg_- cap_ mid		neg_- cap_ dn	offset coarse polarity up	offset coarse polarity mid	offset coarse polarity dn	offset fine polarity up	offset fine polarity mid	offset fine polarity dn					amp_vga							amp_v- ga_ last_ stage				04h											
offset_coarse_up				offset_coarse_mid						offset_coarse_dn					off- set_fine_up			off- set_fine_mid			off- set_fine_dn				08h											
																																0Ch-1Ch				

Table 1099 - [Internal] SLRP - Serdes Lane Receive Parameters Register Fields for USR

Offset	Bits	Name	Description	Access
00h	31:25	mixer00		RW
	24:18	mixer90		RW
	17:14	termination		RW
04h	31:30	neg_cap_up	Negative capacitance for up input buffer	RW
	29:28	neg_cap_mid	Negative capacitance for middle input buffer	RW
	27:26	neg_cap_dn	Negative capacitance for downinput buffer	RW
	25	offset_ coarse_polari- ty_up		RW
	24	offset_ coarse_polari- ty_mid		RW
	23	offset_ coarse_polari- ty_dn		RW
	22	offset_fine_po- larity_up		RW
	21	offset_fine_po- larity_mid		RW
	20	offset_fine_po- larity_dn		RW
	12:8	amp_vga		RW
	4:0	amp_v- ga_last_stage		RW
08h	31:26	offset_coarse_up		RW
	25:20	offset_ coarse_mid		RW
	19:14	offset_coarse_dn		RW
	13:10	offset_fine_up		RW
	9:6	offset_fine_mid		RW
	5:2	offset_fine_dn		RW

28.19.43[Internal] SLRIP - Serdes Lane Receive Input Buffer Parameters

This register enables to read or set the serdes input buffer parameters on the receive side.

When read, the SLRIP returns the receive setting for the input buffer of a specific lane. When the link is up or down the last high speed setting is returned.

Table 1100 - [Internal] Serdes Lane Receive Input Buffer Parameters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
status				version				local_port								pnat				lane												00h		
ib_sel																																		04h
ffe_gain																																08h		
ffe_tap0								ffe_tap1								ffe_tap2								ffe_tap3								0Ch		
ffe_tap4								ffe_tap5								ffe_tap6								ffe_tap7								10h		
ffe_tap8								sel_enc0												ffe_tap_en								14h						
sel_enc1								sel_enc2								mixer_offset0								mixer_offset1								18h		
saved0_sel_enc0								saved0_mixer_offset0								saved1_sel_enc0								saved1_mixer_offset0								1Ch		
																								current_phase								20h		
																																24h-2Ch		

Table 1101 - [Internal] SLRIP - Serdes Lane Receive Parameters Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type: 0 - Local port number 1 - IB Port number 3 - Out of band / PCI	Index
	11:8	lane	physical lane number.	Index

Table 1101 - [Internal] SLRIP - Serdes Lane Receive Parameters Register Fields

Offset	Bits	Name	Description	Access
04h	31:29	ib_sel	input buffer select: 0 - IB0 1 - IB1 2 - IB2 3 - IB3 4 - Active up 5 - Active mid 6 - Active down 7 - Inactive	Index
08h	31:0	ffe_gain	FFE gain calibration	RW
0Ch	31:24	ffe_tap0	9 Taps for input buffer.	RW
	23:16	ffe_tap1		RW
	15:8	ffe_tap2		RW
	7:0	ffe_tap3		RW
10h	31:24	ffe_tap4		RW
	23:16	ffe_tap5		RW
	15:8	ffe_tap6		RW
	7:0	ffe_tap7		RW
14h	31:24	ffe_tap8		RW
	23:16	sel_enc0	Weight for the step multiplication for mixer offset 0.	RW
	8:0	ffe_tap_en		RW
18h	31:24	sel_enc1	Weight for the step multiplication for mixer offset 1.	RW
	23:16	sel_enc2	Weight for the step multiplication for eye scan.	RW
	15:8	mixer_offset0		RW
	7:0	mixer_offset1		RW
1Ch	31:24	saved0_sel_enc0	Tuning value of sel_enc0 (measurement 0)	RW
	23:16	saved0_mixer_offset0	Tuning value of mixer_offset0 (measurement 0)	RW
	15:8	saved1_sel_enc0	Tuning value of sel_enc0 (measurement 1)	RW
	7:0	saved1_mixer_offset0	Tuning value of mixer_offset0 (measurement 1)	RW
20h	7:0	current_phase		RW

28.19.44[Internal] SLRD - Serdes Lane Receive Defect

This register enables to read the serdes receive defect indications.

Table 1102 - SLRD - Serdes Lane Receive Defect Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
status				version				local_port								pnat				lane														00h
																																04h		
					nd_res	clk90_res	clk00_res								failed_lane							eo_err_0	eo_err_1						dp2_res	dp1_res	dp0_res	08h		
clk00_pu_cmh																clk00_pu_cml																0Ch		
clk00_pd_cmh																clk00_pd_cml																10h		
clk90_pu_cmh																clk90_pu_cml																14h		
clk90_pd_cmh																clk90_pd_cml																18h		
nd_pu_cmh																nd_pu_cml																1Ch		
nd_pd_cmh																nd_pd_cml																20h		
																																24h		

Table 1103 - SLRD - Serdes Lane Receive Defect Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 40nm products 1 - 28nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	lane	lane number.	Index

Table 1103 - SLRD - Serdes Lane Receive Defect Register Fields

Offset	Bits	Name	Description	Access
08h	26	nd_res		RO
	25	clk90_res		RO
	24	clk00_res		RO
	16	failed_lane		RO
	9	eo_err_0		RO
	8	eo_err_1		RO
	2	dp2_res		RO
	1	dp1_res		RO
	0	dp0_res		RO
0Ch	31:16	clk00_pu_cmh		RO
	15:0	clk00_pu_cml		RO
10h	31:16	clk00_pd_cmh		RO
	15:0	clk00_pd_cml		RO
14h	31:16	clk90_pu_cmh		RO
	15:0	clk90_pu_cml		RO
18h	31:16	clk90_pd_cmh		RO
	15:0	clk90_pd_cml		RO
1Ch	31:16	nd_pu_cmh		RO
	15:0	nd_pu_cml		RO
20h	31:16	nd_pd_cmh		RO
	15:0	nd_pd_cml		RO

28.19.45SLTP - Serdes Lane Transmit Parameters

This register enables to read or set the serdes transmit parameters.

The register shall be read after write as the configuration might vary according to device capabilities.

Table 1104 - SLTP - Serdes Lane Transmit Parameters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status				version				local_port								pnat		qg		lane				lane_speed							c_db	00h				
page_data																																04h-18h				

Table 1105 - SLTP - Serdes Lane Transmit Parameters Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 40nm products 1 - 28nm products 3 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	13	gb	[Internal]: Set to '1' for Gearbox control and access. When set, see the relevant <i>page_data</i> .	OP
	11:8	lane	Physical lane number	Index
	7:4	lane_speed	Reserved for PCIe. The lane speed for TX settings: 0 - SDR / Gen1 (PCIe) 1 - DDR / Gen2 (PCIe) 2 - QDR / Gen3 (PCIe) 3 - FDR10 (10GE / 40GE) 4 - FDR (56GE) / Gen4 (PCIe) 5 - EDR (25GE / 50GE / 100GE) / Gen5 (PCIe) 6 - HDR (50GE / 200GE / 400GE) 7-9 - Reserved 10 - 1GE 11 - 2.5GE (XAUI) 12 - 50GE-KR4 13-15 - Reserved	RW
	0	c_db	copy transmitter parameters to Data Base. [internal] The tx parametrs will be copied according to <i>lane_speed</i> , <i>module_type</i> , <i>attenuation</i> . The <i>module_type</i> (Copper, Active or Backplane) and <i>attenuation</i> are taking from the module.	OP
04h-18h	192	page_data	See Table 1106, "SLTP - Serdes Lane Transmit Parameters Register Layout for 28nm and 40nm," on page 1304 See Table 1108, "SLTP - Serdes Lane Transmit Parameters Register Layout for 16nm," on page 1305 See Table 1110, "[Internal] Serdes Lane Transmit Parameters Register Layout For Gearbox," on page 1306	RO

Table 1106 - SLTP - Serdes Lane Transmit Parameters Register Layout for 28nm and 40nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
							polarity	ob_tap0				ob_tap1				ob_tap2																04h
								ob_leva				ob_pre-emp_mode				ob_reg				ob_bias												08h
ob_norm	ob_bad_stat																															0Ch

Table 1107 - SLTP - Serdes Lane Transmit Parameters Register Fields for 28nm and 40nm

Offset	Bits	Name	Description	Access
00h	31:0	reserved		RO
04h	24	polarity	Tx lane polarity	RW
	23:16	ob_tap0	[Internal] In case of SET operation FW will normalize according to the device rules (if need and possible). If SerDes configuration succeed, ob_bad_stat shall be cleared. If Normalization was performed during SerDes configuration ob_norm shall be set. In case of GET operation only the value that is actually configured on the SerDes (i.e. normalized) will be return. [Internal] For PCIE ports can be only query (RO)	RW
	15:8	ob_tap1		RW
	7:0	ob_tap2		RW
08h	23:20	ob_leva		RW
	19:16	ob_preemp_mode	Concatenation of taps polarity + preemp_sel	RW
	15:8	ob_reg	(ref_txcm1100_reg)	RW
	7:0	ob_bias		RW
0Ch	31	ob_norm	OB Normalization status: 0 - No Normalization was done 1 - Normalization performed	RO
	30:29	ob_bad_stat	Bitmask for bad SET status: Bit 0 - Illegal ob_bais value Bit 1 - Illegal ob taps polarity (distance between taps)	RO

Table 1108 - SLTP - Serdes Lane Transmit Parameters Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
pre_2_tap								pre_tap								main_tap								post_tap								04h
ob_m2lp																ob_amp											ob_alev_out				08h	
ob_bad_stat																																0Ch
obplev								obnlev								regn_bfm1p								regp_bfm1n								10h
blev																tx_alev					alev_plus_bfm2				alev_minus_bfm2						14h	

Table 1109 - SLTP - Serdes Lane Transmit Parameters Register Fields for 16nm

Offset	Bits	Name	Description	Access
04h	31:24	pre_2_tap	Value of each tap is between -96 to 96 NOTE: The total sum of all absolute taps values should be 96 or smaller. In case of a 'read' operation, a normalized value is returned. For PCIe ports, only 'read' operation is available (RO).	RW
	23:16	pre_tap		RW
	15:8	main_tap		RW
	7:0	post_tap		RW
08h	31:25	ob_m2lp	MSB to LSB proportion. Values can be set between -10 to 50. MSB emphasis achieved when value is between -10 to -1 LSB emphasis achieved when value is between 1 to 50. When set to 0 - equal proportion.	RW
	14:8	ob_amp	Output amplitude. Values can be set in the range of (10-100).	RW
	4:0	ob_alev_out	Output common mode. Values can be set in the range of (0-30)	RW
0Ch	31:16	ob_bad_stat	Bitmask for bad 'set' status: 0: configuration_warning - taps values can't be set to serdes due to internal limitations. Actual TX configuration is modify internally with the same sum of taps weight. 11: Illegal_ob_combination - ob_m2lp, ob_amp and ob_alev is in legal range each, but the total combination can't be set to serdes due to internal limitations, the Set command is ignored. 12: Illegal_ob_m2lp 13: Illegal_ob_amp 14: Illegal_ob_alev_out 15: Illegal_taps Bit 12-0 - Reserved /Unknown	RO

Table 1109 - SLTP - Serdes Lane Transmit Parameters Register Fields for 16nm

Offset	Bits	Name	Description	Access
10h	31:24	obplev	[Internal]: Values can be set in the range of (0-200).	RW
	23:16	obnlev	[Internal]: Values can be set in the range of (0-200).	RW
	15:8	regn_bfm1p	[Internal]: Values can be set in the range of (0-200).	RW
	7:0	regp_bfm1n	[Internal]: Values can be set in the range of (0-200).	RW
14h	31:25	blev		RW
	14:10	tx_alev		RO
	9:5	alev_plus_bfm2	[Internal]: Values can be set in the range of (0-28).	RO
	4:0	alev_minus_bfm2	[Internal]: Values can be set in the range of (0-28).	RO

Table 1110 - [Internal] Serdes Lane Transmit Parameters Register Layout For Gearbox

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
main_tap																pre_tap																00h
																post_tap																04h
inner_eye_2																inner_eye_1																08h
																									tx_swing			0Ch				
																														10h-14h		

Table 1111 - [Internal] SLTP - Serdes Lane Transmit Parameters Register Fields For Gearbox

Offset	Bits	Name	Description	Access
00h	31:16	main_tap		RW
	15:0	pre_tap		RW
04	15:0	post_tap		RW
08h	31:16	inner_eye_2		RW
	15:0	inner_eye_1		RW
0C	3:0	tx_swing		RW

28.19.46[Internal] SLUTP - Serdes Lane USR Transmit Parameters

This register enables to read or set USR serdes transmit parameters.

The register shall be read after write as the configuration might vary according to device capabilities.

Table 1112 - SLUTP - Serdes Lane USR Transmit Parameters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status				version				local_port								pnat				lane				port_type							c_db	00h				
ob_m2lp																ob_amp														ob_resolution		04h				
ob_bad_stat																																				08h

Table 1113 - SLUTP - Serdes Lane USR Transmit Parameters Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	3 - 16nm products 4 - 7nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	lane	Physical lane number	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 4 - USR Main 5 - USR Tile Other values are reserved.	Index
	0	c_db	copy transmitter parameters to Data Base. [internal] The tx parametrs will be copied according to USR data-base entry	OP

Table 1113 - SLUTP - Serdes Lane USR Transmit Parameters Register Fields

Offset	Bits	Name	Description	Access
04h	31:25	ob_m2lp	[Internal]: MSB to LSB proportion. Values can be set in the range of (0-70). MSB emphasis achieved when set in the range of (0-9) and LSB emphasis when in the range of (11-70). When set to '10' - neutral.	RW
	14:8	ob_amp	[Internal]: Output amplitude. Values can be set in the range of (20-120). Minimal amplitude is 100mV for the value of 20 while the maximal amplitude is 600mV for a value set to 120.	RW
	1:0	ob_resolution	[Internal]: Output resolution mode. Values can be set in the range of (0-3)	RW
08h	31:28	ob_bad_stat	Bitmask for bad 'set' status: Bit 0 - Illegal ob_m2lp Bit 1 - Illegal ob_amp Bit 2 - Illegal ob_resolution	RO

28.19.47SLRG - Serdes Lane Receive Grade

This register enables to read the serdes receive grade.

Table 1114 - SLRG - Serdes Lane Receive Grade Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
status				version				local_port								pnat				lane													00h				
page_data																																					04h-24h

Table 1115 - SLRG - Serdes Lane Receive Grade Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 40nm products 1 - 28nm products 3 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	lane	Physical lane number	Index
04h-24h	288	page_data	See Table 1116, “SLRG - Serdes Lane Receive Grade Register Layout for 40nm and 28nm,” on page 1310 See Table 1118, “SLRG - Serdes Lane Receive Grade Register Layout for 16nm,” on page 1312	RO

Table 1116 - SLRG - Serdes Lane Receive Grade Register Layout for 40nm and 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																grade_lane_speed	00h
grade_version								grade																								04h	
off-set_u_nits	phase_unit_s		height_grade_type					height_grade																								08h	
height_dz																height_dv																0Ch	

Table 1116 - SLRG - Serdes Lane Receive Grade Register Layout for 40nm and 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																height_sigma																10h
height_eo_pos																height_eo_neg																14h
				phase_grade_type		phase_grade																								18h		
								phase_eo_pos																phase_eo_neg								1Ch
ffe_set_tested																test_errors_per_lane																20h

Table 1117 - SLRG - Serdes Lane Receive Grade Register Fields for 40nm and 28nm

Offset	Bits	Name	Description	Access
00h	3:0	grade_lane_speed	The lane speed on which the grade was measured: 0 - SDR 1 - DDR 2 - QDR 3 - FDR10 (10GE / 40GE) 4 - FDR (56GE) 5 - EDR (25GE / 50GE / 100GE) 6 - 9 - reserved 10 - 1GE 11 - 2.5GE (XAUI) 12 - 50GE-KR4 13 - 15 - reserved	RO
04h	31:24	grade_version		RO
	23:0	grade		RO
08h	31:30	offset_units	Determine the units of <i>height_eo_pos</i> and <i>height_eo_neg</i> : 0 - mV after slicer (including A2D sample) 1 - mV before slicer	RW
	29:28	phase_units	Determine the units of <i>phase_eo_pos</i> and <i>phase_eo_neg</i> : 0 - psec 1 - ticks	RW
	27:24	height_grade_type	0 - BER Bath 1 - 1s ratio 2 - IET 3 - XOR	RO
	23:0	height_grade		RO

Table 1117 - SLRG - Serdes Lane Receive Grade Register Fields for 40nm and 28nm

Offset	Bits	Name	Description	Access
0Ch	31:16	height_dz	When height_grade_type = EO holds height_eo_pos	RO
	15:0	height_dv	N/A	RO
10h	15:0	height_sigma		RO
14h	31:16	height_eo_pos	Voltage offset from eye center to the positive side.	RO
	15:0	height_eo_neg	Voltage offset from eye center to the negative side.	RO
18h	27:24	phase_grade_type	0 - Reserved 1 - Reserved 2 - EO	RO
	23:0	phase_grade		RO
1Ch	23:16	phase_eo_pos	The offset from the eye center to the positive side.	RO
	7:0	phase_eo_neg	The offset from the eye center to the negative side.	RO
20h	31:16	ffe_set_tested	The number of space search steps performed during the lane tuning. 0 - N/A - no search was performed.	RO
	15:0	test_errors_per_lane	Number of bit errors measured after the tuning has completed (BER). The time window for the test is defined in PTAS “ber_test_time”.	RO

Table 1118 - SLRG - Serdes Lane Receive Grade Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												grade_lane_speed	00h			
grade_version								grade																				04h				
height_eo_pos_up														height_eo_neg_up														08h				
phase_eo_pos_up								phase_eo_neg_up								height_eo_pos_mid												0Ch				
height_eo_neg_mid														phase_eo_pos_mid								phase_eo_neg_mid						10h				
height_eo_pos_low														height_eo_neg_low														14h				
phase_eo_pos_low								phase_eo_neg_low									off-set_u nits	phase _unit s	height_grade _type				phase_grade _type				18h					

Table 1119 - SLRG - Serdes Lane Receive Grade Register Fields for 16nm

Offset	Bits	Name	Description	Access
00h	3:0	grade_lane_speed	The lane speed on which the grade was measured: 0 - SDR / PCIe Gen 1 1 - DDR / PCIe Gen 2 2 - QDR / PCIe Gen 3 3 - FDR10 (10GE / 40GE) 4 - FDR (56GE) / PCIe Gen 4 5 - EDR (25GE / 50GE / 100GE) / PCIe Gen 5 6 - HDR (50GE / 200GE / 400GE) 7- 9 - reserved 10 - 1GE 11 - 2.5GE (XAUI) 12 - 50GE-KR4 13 - 15 - reserved	RO
04h	31:24	grade_version		RO
	23:0	grade	The grade that the lane received based on a specific configuration. <i>[Internal]: returns the minimal grade from the 3 grades available (upper_grade, mid_grade, lower_grade)</i>	RO
08h	31:16	height_eo_pos_upper	Voltage offset from eye center to the positive side of the upper eye. Only applicable for PAM4 signal.	RO
	15:0	height_eo_neg_upper	Voltage offset from eye center to the negative side of the upper eye. Only applicable for PAM4 signal.	RO
0Ch	31:24	phase_eo_pos_upper	The offset from the eye center to the positive (right) side of the upper eye. Only applicable for PAM4 signal.	RO
	23:16	phase_eo_neg_upper	The offset from the eye center to the negative (left) side of the upper eye. Only applicable for PAM4 signal.	RO
	15:0	height_eo_pos_mid	Voltage offset from eye center to the positive side of the middle eye.	RO
10h	31:16	height_eo_neg_mid	Voltage offset from eye center to the negative side of the middle eye.	RO
	15:8	phase_eo_pos_mid	The offset from the eye center to the positive (right) side of the middle eye.	RO
	7:0	phase_eo_neg_mid	The offset from the eye center to the negative (left) side of the middle eye.	RO
14h	31:16	height_eo_pos_lower	Voltage offset from eye center to the positive side of the lower eye. Only applicable for PAM4 signal.	RO
	15:0	height_eo_neg_lower	Voltage offset from eye center to the negative side of the lower eye. Only applicable for PAM4 signal.	RO

Table 1119 - SLRG - Serdes Lane Receive Grade Register Fields for 16nm

Offset	Bits	Name	Description	Access
18h	31:24	phase_eo_pos_low	The offset from the eye center to the positive (right) side of the lower eye. Only applicable for PAM4 signal.	RO
	23:16	phase_eo_neg_low	The offset from the eye center to the negative (left) side of the lower eye. Only applicable for PAM4 signal.	RO
	11:10	offset_units	Determine the units of <i>height_eo_pos</i> and <i>height_eo_neg</i> : 0 - mV after slicer (including A2D sample) 1 - mV before slicer	RW
	9:8	phase_units	Determine the units of <i>phase_eo_pos</i> and <i>phase_eo_neg</i> : 0 - psec 1 - ticks	RW
	7:4	height_grade_type	0 - BER Bath 1 - 1s ratio 2 - IET (Idle Error Traffic) 3 - XOR	RO
	3:0	phase_grade_type	0 - Reserved 1 - Reserved 2 - EO	RO

28.19.48 SLREG - SerDes Lane Receive Eye Grade (Internal)

This register enables to read the SerDes receive grade for each of the eyes. In contrary to SLRG, SLREG register provides internal and more detailed information about the eye grades.

Table 1120 - SLREG - Serdes Lane Receive Eye Grade Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status				version				local_port				pnat				lane				port_type						eidx		00h					
page_data																																04h-48h	

Table 1121 - SLREG - Serdes Lane Receive Grade Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid Note: Links that do not perform tuning return <i>invalid</i>. i.e. SDR/DDR/Mloop	RO
	27:24	version	0 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number 3 - Out of band / PCI	Index
	11:8	lane	physical lane number.	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index
	1:0	eidx	Eye Index. 0 - Upper 1 - Mid 2 - Lower	Index

Table 1121 - SLREG - Serdes Lane Receive Grade Register Fields

Offset	Bits	Name	Description	Access
04h-48h	576	page_data	See Table 1122, “SLREG - Serdes Lane Receive Eye Grade Register Layout for 16nm,” on page 1316 See Table 1124, “SLREG - Serdes Lane Receive Grade Register Layout for USR,” on page 1318	RO

Table 1122 - SLREG - Serdes Lane Receive Eye Grade Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												grade_lane_speed	00h			
								grade																				04h				
								height_grade																				08h				
								phase_grade																				0Ch				
initial_error																												10h				
post_error																												14h				
initial_area																												18h				
post_area																												1Ch				
post_area_centered																												20h				
gd_iter_num_errors														gd_iter_num_area														24h				
				post_sel_enc_slave																		initial_sel_enc_slave										28h
				rect_size																		post_sel_enc_slave_centered										2Ch
																	sel_enc_area_fault_count						zero_area_count						30h			

Table 1123 - SLREG - Serdes Lane Receive Grade Register Fields for 16nm

Offset	Bits	Name	Description	Access
00h	3:0	grade_lane_speed	The lane speed on which the grade was measured: 0 - SDR 1 - DDR 2 - QDR 3 - FDR10 (10GE / 40GE) 4 - FDR (56GE) 5 - EDR (25GE / 50GE / 100GE) 6 - HDR (50GE / 200GE / 400GE) 7 - 9 - reserved 10 - 1GE 11 - 2.5GE (XAUI) 12 - 50GE-KR4 13 - 15 - reserved	RO
04h	23:0	grade	Eye grade	RO
08h	23:0	height_grade		RO
0Ch	23:0	phase_grade		RO
10h	31:0	initial_error		RO
14h	31:0	post_error		RO
18h	31:0	initial_area		RO
1Ch	31:0	post_area		RO
20h	31:0	post_area_centered		RO
24h	31:16	gd_iter_num_errors		RO
	15:0	gd_iter_num_area		RO
28h	27:16	post_sel_enc_slave		RO
	11:0	initial_sel_enc_slave		RO
2Ch	27:16	rect_size		RO
	11:0	post_sel_enc_slave_centered		RO

Table 1123 - SLREG - Serdes Lane Receive Grade Register Fields for 16nm

Offset	Bits	Name	Description	Access
30h	15:8	sel_enc_area - fault_count		RO
	7:0	zero_area_count	The number of configuration sets that were tested on the central eye before the split to 3 eyes.	RO

Table 1124 - SLREG - Serdes Lane Receive Grade Register Layout for USR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h
																																14h
																																18h
																																1Ch
																																20h
																																24h
																																28h
																																2Ch
																																30h
																																34h
																																38h
																																3Ch
																																40h
																																44h

Table 1125 - SLREG - Serdes Lane Receive Grade Register Fields for USR

Offset	Bits	Name	Description	Access
00h	28:0	Grade	Grade value	RO
04h	31:20	phase_bounds		RO
	17:8	eye_symmetry_command	eye symmetry correction command sent to transmitter	RO
	5:0	optimal_phase		RO
08h	31:16	amp_phase0		RO
	15:0	amp_phase1		RO
0Ch	31:16	amp_phase2		RO
	15:0	amp_phase3		RO
10h	31:16	amp_phase4		RO
	15:0	amp_phase5		RO
14h	31:16	amp_phase6		RO
	15:0	amp_phase7		RO
18h	31:16	amp_phase8		RO
	15:0	amp_phase9		RO
1Ch	31:16	amp_phase10		RO
	15:0	amp_phase11		RO
20h	31:16	amp_phase12		RO
	15:0	amp_phase13		RO
24h	31:16	amp_phase14		RO
	15:0	amp_phase15		RO
28h	31:16	amp_phase16		RO
	15:0	amp_phase17		RO
2Ch	31:16	amp_phase18		RO
	15:0	amp_phase19		RO
30h	31:16	amp_phase20		RO
	15:0	amp_phase21		RO

Table 1125 - SLREG - Serdes Lane Receive Grade Register Fields for USR

Offset	Bits	Name	Description	Access
34h	31:16	amp_phase22		RO
	15:0	amp_phase23		RO
38h	31:16	amp_phase24		RO
	15:0	amp_phase25		RO
3Ch	31:16	amp_phase26		RO
	15:0	amp_phase27		RO
40h	31:16	amp_phase28		RO
	15:0	amp_phase29		RO
44h	31:16	amp_phase30		RO
	15:0	amp_phase31		RO

28.19.49SLRED - SerDes Lane Receive Eye Diagram

This register enables to read the SerDes receive eye diagram for each eye

Table 1126 - *SLRED - Serdes Lane Receive Eye Diagram Register Layout*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				<u>version</u>				<u>local_port</u>								<u>pnat</u>				<u>lane</u>			<u>port_type</u>			<u>page_data_sel</u>				00h		
en	no	err_re	eye_di- ag_dim	measure_ time		ps_ticks								mv_ticks						no	status	lane_speed				04h						
n	de	s_ma																									ntissa	n_de struc t_ctl cap				
<u>page_data</u>																																08h-8Ch

Table 1127 - *SLRED - Serdes Lane Receive Eye Diagram Register Fields*

<u>Offset</u>	<u>Bits</u>	<u>Name</u>	<u>Description</u>	<u>Access</u>
00h	27:24	<u>version</u>	0 - 16nm products	RO
	23:16	<u>local_port</u>	Local port number.	Index
	15:14	<u>pnat</u>	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB / label port number 3 - Out of band / PCI	Index
	11:8	<u>lane</u>	physical lane number.	Index
	7:4	<u>port_type</u>	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index

Table 1127 - *SLRED - Serdes Lane Receive Eye Diagram Register Fields*

Offset	Bits	Name	Description	Access
00h	3:0	<u>page_data_sel</u>	select page data number to display. 0 - page data 0 (indexes 0 to 399) 1 - page data 1 (indexes 400 to 799) 2 - page data 2 (indexes 800 to 1199) 3 - page data 3 (indexes 1200 to 1599) 4 - page data 4 (indexes 1600 to 1999) 5 - page data 5 (indexes 2000 to 2399)	<u>Index</u>
04h	31	<u>en</u>	<u>start eye scan</u>	<u>WO</u>
	30	<u>non_destruct_c- trl</u>	0 - non destructive scan 1 - destructive scan Note: if non destructive scan is not supported this field is ignored and a destructive scan will be operated	<u>RW</u>
	29:28	<u>err_res_mantissa</u>	mantissa resolution for err_val. err value will represent errors between- $err\_res\_mantissa^{err\_val} + 1$ to $err\_res\_mantissa^{err\_val}$ 0 - err_res_mantissa is 10 1-3 - reserved For example if err_res_mantissa is 10: 0 err_value represents 0 errors 1 err_value represents 1-10 errors 2 err_value represents 11-100 errors 3 err_value represents 101-1000 errors ... 7 err_value represents $10^6 + 1 - 10^7$ errors	<u>RW</u>
	27:25	<u>eye_diag_dim</u>	dimensions of eye scan matrix in the format of <i>hight_dim</i> x <i>width_dim</i> 0 - 20x20 (400 err val) 1 - 32x32 (1024 err val) 2 - 32x64 (2048 err val) 3 - 64x32 (2048 err val)	<u>RW</u>
	24:21	<u>measure_time</u>	max measure time in sec for all diagram. 0 - 10 sec 1 - 30 sec 2 - 60 sec 3 - 90 sec 4 - 120 sec 5 - 240 sec 6 - 480 sec 7 - 600 sec 8 - 900 sec 9-15 reserved each tic measure time will be $measure\_time / (hight\_dim \times width\_dim)$	<u>RW</u>
	20:14	<u>ps_ticks</u>		<u>RO</u>
	13:7	<u>mv_ticks</u>		<u>RO</u>

Table 1127 - *SLRED - Serdes Lane Receive Eye Diagram Register Fields*

Offset	Bits	Name	Description	Access
04h	<u>6</u>	<u>non_destruct_ - cap</u>	<u>if '1' non destructive scan is supported</u>	<u>RO</u>
	<u>5:4</u>	<u>status</u>	<u>eye scan status:</u> <u>0 - eye scan was not performed.</u> <u>1 - eye scan pending to execute.</u> <u>2 - Performing eye scan.</u> <u>3 - eye scan completed.</u>	<u>RO</u>
	<u>3:0</u>	<u>lane_speed</u>	<u>The lane speed on which the eye scan was measured</u> <u>0 - SDR (2.5 Gb/s) / Gen1 (PCIe)</u> <u>1 - DDR (5 Gb/s) / Gen2 (PCIe)</u> <u>2 - QDR (10 Gb/s) / Gen3 (PCIe)</u> <u>3 - FDR10 (10GE / 40GE) (10.3125 Gb/s)</u> <u>4 - FDR (56GE) (14.0625 Gb/s) / Gen4 (PCIe)</u> <u>5 - EDR (25GE / 50GE / 100GE) (25.78125 Gb/s) / Gen5 (PCIe)</u> <u>6 - HDR (50GE / 200GE / 400GE) (26.5625Gbd / 53.125Gb/s)</u> <u>7 - 9 - reserved</u> <u>10 - 1GE (1.25 Gb/s)</u> <u>11 - 2.5GE (XAUI) (3.125 Gb/s)</u> <u>12 - 50GE-KR4 (12.89 Gb/s)</u> <u>13 - 15 - reserved</u>	<u>RO</u>
<u>08h-8Ch</u>	<u>1088</u>	<u>page_data</u>	<u>page data for index <i>page_data_sel</i></u> <u>the page data will contain the err_val for each index in ascending order.</u> <u>Each err_val[i] represents the bit errors of row floor(i / width_dim) and column i mod width_dim</u> <u>each err_val indicates the number of bit errors for one tick</u>	<u>RO</u>

Table 1128 - *SLRED - Serdes Lane Receive Eye Diagram Register Layout for page_data*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
err_val[0]				err_val[1]				err_val[2]				err_val[3]				err_val[4]				err_val[5]				err_val[6]				err_val[7]				00h
...																																...
err_val[312]				err_val[313]				err_val[314]				err_val[315]				err_val[316]				err_val[317]				err_val[318]				err_val[319]				9Ch

Table 1129 - *SLRED - Serdes Lane Receive Eye Diagram Register Fields for page_data*

Offset	Bits	Name	Description	Access
00h-9Ch	31:28	err_val[320]	each err_val is 4 bits in ascending order. each page contains 320 err values.	RO

28.19.50 SLCCT - SerDes Lane CTLE Calibration Table Register

This register enables to query the CTLE calibration lookup table up to 20 different configurations.

Table 1130 - SLCCT - Serdes Lane CTLE Calibration Table Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status				version				local_port								pnat				lane												00h
																				num_of_idx							conf_index				04h	
ctle_calib_set[0]																																08h
ctle_calib_set[1]																																0Ch
ctle_calib_set[2]																																10h
ctle_calib_set[3]																																14h
ctle_calib_set[4]																																18h

Table 1131 - SLCCT - Serdes Lane CTLE Calibration Table Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	status	0 - Invalid 1 - Valid	RO
	27:24	version	0 - 16nm products	RO
	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number 3 - Out of band / PCI	Index
	11:8	lane	physical lane number.	Index
04h	11:8	num_of_idx	number of configuration sets indices to access	RW
	4:0	conf_index	Start index for the configuration set.	RW
08h-18h	31:0	ctle_calib_set[5]	See Table 1140, "PLL Status Data Layout for 28nm," on page 1331	RO

Table 1132 - CTLE Calibration Set Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				bgn_offset_p1								bgn_offset_n1								bgn_offset_p0								bgn_offset_n0				00h

Table 1133 - CTLE Calibration Set Fields

Offset	Bits	Name	Description	Access
00h	28:24	bgn_offset_p1	Offset of biasgen_p1	RO
	20:16	bgn_offset_n1	Offset of biasgen_n1	RO
	12:8	bgn_offset_p0	Offset of biasgen_p0	RO
	4:0	bgn_offset_n0	Offset of biasgen_n0	RO

28.19.51[Internal] PTAS - Phy Tuning Algorithm Settings

This register enables to read or set the tuning algorithm parameters.

Table 1134 - PTAS - Phy Tuning Algorithm Settings Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												00h				
algorithm_options																				repetitions_-mode		num_of_repetitions						04h				
grade_version								height_grade_type				phase_grade_type				height_grade_weight						phase_grade_weight						08h				
gisim_measure_bits																adaptive_tap_measure_bits												0Ch				
ber_bath_high_error_threshold																ber_bath_mid_error_threshold												10h				
ber_bath_low_error_threshold																one_ratio_high_threshold												14h				
one_ratio_high_mid_threshold																one_ratio_low_mid_threshold												18h				
one_ratio_low_threshold																ndeo_error_threshold												1Ch				
mixer_offset_step_size																						mix90_phase_for_volt-age_bath						20h				
mixer_offset_start																mixer_offset_end												24h				
																				ber_test_time						28h						

Table 1135 - PTAS - Phy Tuning Algorithm Settings Register Fields

Offset	Bits	Name	Description	Access
00h	31:0	reserved		RO
04h	31:16	algorithm_options	Bit 0 - calib_for_each_conf_en When set perform calibration for each set Bit 1- adaptive_tap_offset_en When set perform adaptive tap tuning, when cleared use a pre-defined tap offset. Bit 2 - phase_measuring_en When set perform also phase measuring as part of the tuning grade.	RW
	11:8	repetitions_-mode	0 - Average 1 - Minimum 2 - Median	RW
	7:0	num_of_repetitions	Number height grade measurement repetitions	RW

Table 1135 - PTAS - Phy Tuning Algorithm Settings Register Fields

Offset	Bits	Name	Description	Access
08h	31:24	grade_version		RW
	23:20	height_grade_type	0 - BER Bath 1 - 1s ratio 2 - EO 3 - 3 - 1s ratio + NDEO	RW
	19:16	phase_grade_type	0 - Reserved 1 - Reserved 2 - EO	RW
	15:8	height_grade_weight		RW
	7:0	phase_grade_weight		RW
0Ch	31:16	gisim_measure_bits		RW
	15:0	adaptive_tap_measure_bits		RW
10h	31:16	ber_bath_high_error_threshold		RW
	15:0	ber_bath_mid_error_threshold		RW
14h	31:16	ber_bath_low_error_threshold		RW
	15:0	one_ratio_high_threshold		RW
18h	31:16	one_ratio_high_mid_threshold		RW
	15:0	one_ratio_low_mid_threshold		RW
1Ch	31:16	one_ratio_low_threshold		RW
	15:0	ndeo_error_threshold		RW

Table 1135 - PTAS - Phy Tuning Algorithm Settings Register Fields

Offset	Bits	Name	Description	Access
20h	31:16	mixer_offset - step_size		RW
	7:0	mix90_phase - for_voltage_bath		RW
24h	31:16	mixer_off- set_start		RW
	15:0	mixer_off- set_end		RW
28h	10:0	ber_test_time	BER test time in 1msec units.	RW

28.19.52[Internal] PPLL - Phy PLL Status Register

This register enables to read the PLL status. Including all the PLLs in the device, network and PCI)

Table 1136 - PPLL - Phy PLL Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
num_pll_groups								pll_group												num_plls								version				00h
page_data																																04h-24h

Table 1137 - PPLL - Phy PLL Status Register Fields

Offset	Bits	Name	Description	Access
00h	31:24	num_pll_groups	number of PLL groups in the device	RO
	23:16	pll_group	PLL group index	Index
	11:8	num_plls	number of PLL in the PLL group	RO
	3:0	version	0 - 28nm products 1 - 16nm products	RO
04h-24h	288	page_data	See Table 1138, “PPLL - Phy PLL Status Register Layout for 28nm,” on page 1330 See Table 1142, “PPLL - Phy PLL Status Register Layout for 16nm,” on page 1332	RO

Table 1138 - PPLL - Phy PLL Status Register Layout for 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															ac	00h
pll_status[0]																																04h-08h
pll_status[1]																																0Ch-10h
pll_status[2]																																14h-18h

Table 1138 - PPLL - Phy PLL Status Register Layout for 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pll_status[3]																																1 Ch-20h

Table 1139 - PPLL - Phy PLL Status Register Fields for 28nm

Offset	Bits	Name	Description	Access
00h	0	ae	Varactors calibration flow was executed	RO
04h-20h	64	pll_status[4]	See Table 1140, “PLL Status Data Layout for 28nm,” on page 1331	RO

Table 1140 - PLL Status Data Layout for 28nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
	lock_cal	lock_status				algo_f_ctrl										analog_algo_num_var							f_ctrl_measure										00h
		analog_var								high_var								low_var								mid_val						04h	

Table 1141 - PLL Status Data Fields for 28nm

Offset	Bits	Name	Description	Access
00h	30	lock_cal	PLL lock after the varactor calibration flow (locked by f_ctrl measure)	RO
	29:28	lock_status	PLL Lock status causes	RO
	25:16	algo_f_ctrl	fctrl value of the varactor algorithm	RO
	15:10	analog_algo_num_var	Number of varactors connected to the analog by algorithm configuration	RO
	9:0	f_ctrl_measure	Measurement of fctrl in ticks Shell be set to 0 if not implemented	RO
04h	29:24	analog_var	Number of varactors connected to the analog	RO
	21:16	high_var	Number of varactors connected to the high voltage	RO
	13:8	low_var	Number of varactors connected to the low voltage	RO
	5:0	mid_val	Number of varactors connected to the configurable voltage	RO

Table 1142 - PPLL - Phy PLL Status Register Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
pll_status[0]																																04h-08h
pll_status[1]																																0Ch-10h
pll_status[2]																																14h-18h
pll_status[3]																																1Ch-20h

Table 1143 - PPLL - Phy PLL Status Register Fields for 16nm

Offset	Bits	Name	Description	Access
04h-20h	64	pll_status[4]	See Table 1144, “PLL Status Data Layout for 16nm,” on page 1332	RO

Table 1144 - PLL Status Data Layout for 16nm

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lock_status		pll_pwrup	lock_pulse		lock_lost_counter																		ac	clear_cause	lock_clk_val	plllock_clk_val	clock_valid	pll_speed	00h			
lock	analog_var			high_var			low_var			mid_val			fctrl_measure																04h			

Table 1145 - PLL Status Data Fields for 16nm

Offset	Bits	Name	Description	Access
00h	31:30	lock_status	PLL Lock status indications. 2 “sticky bits”. Bit 31 - set to ‘1’ in case of lock Bit 30 - set to ‘1’ in case of loss of lock	RO
	29	pll_pwrup	If set, PLL is powered on (while working on specific speed, all other unused PLLs are powered off) Note: For PCIe, several PLLs can work in parallel.	RO
	28:25	lock_pulse	Sensitivity of the lock loss indication that reflects the <i>lock_lost_counter</i> .	RW
	24:9	lock_lost_counter	Counter of lock losses To clear, write ‘0’.	RW
	8	ae	Varactors calibration algorithm was executed if set to ‘1’	RO
	7	clear_cause	Clears the <i>lock_status</i> and <i>lock_clk_val_cause</i> fields.	WO
	6:5	lock_clk_val_cause	2 “sticky bits”. Bit 6 - Clock was valid and PLL lock was accomplished. Bit 5 - Clock was not valid for some time or loss of lock.	RO
	4	plllock_clk_val	Real time indication of PLL lock and a valid clock.	RO
	3	clock_valid	Clock is valid	RO
04h	2:0	pll_speed	The frequency that the PLL is set to. 0 - SDR / QDR / 1GE for Ethernet / Gen1 (PCIe) 1 - FDR / Gen2 (PCIe) 2 - EDR / 25GE for Ethernet / Gen3 (PCIe) 3 - HDR / 26GE for Ethernet 4 - 10GE for Ethernet / Gen4 (PCIe) 5 - Gen5 (PCIe)	RO
	31	lock	Real time indication of PLL lock.	RO
	30:26	analog_var	Number of varactors connected to the analog (6-8 typical number)	RO
	25:21	high_var	Number of varactors connected to the high voltage	RO
	20:16	low_var	Number of varactors connected to the low voltage	RO
	15:10	mid_val	Number of varactors connected to the configurable voltage	RO
	9:0	fctrl_measure	Measurement of fctrl in ticks Shell be set to 0 if not implemented	RO

28.19.53 PHBR - Ports Histogram Binding Register

The PHBR binds histogram agents per port.

Table 1146 - PHBR - Port Histogram Binding Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode								local_port								pnat																	00h
																mode_cap								hist_id								04h	
	mode															hist_type															08h		
hist_parameters																																0Ch	
hist_min_value																																10h	
hist_max_value																																14h	
																												sample_time				18h	

Table 1147 - PHBR - Port Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	opcode	Opcode: 0 - Bind a <i>local_port</i> , <i>hist_id</i> 1 - Unbind a <i>local_port</i> , <i>hist_id</i>	RW
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type: 0 - Local port number 1 - IB port number	Index
04h	15:14	mode_cap	Mode capability bit 0 - Exponential histogram supported	RO
	7:0	hist_id	Histogram id to bind from 0 to <i>cap_max_port_hist_id - 1</i>	Index
08h	30:28	mode	Mode: 0 - Linear histogram 1 - Exponential histogram	RW
	15:0	hist_type	0x0020: PortRcvData 0x0021: PortRcvData_Prio 0x0040: PortXmitData 0x0041: PortXmitData_Prio 0x0042: PortXmitWait	RW

Table 1147 - PHBR - Port Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	hist_parameters	According to <i>hist_type</i> See Table 1148, “Histograms Hist-type,” on page 1335	RW
10h	31:0	hist_min_value	Min value for histogram main range Units according to the units of the <i>hist_type</i> must be greater than zero.	RW
14h	31:0	hist_max_value	Max value for histogram main range NOTE: for Spectrum device the following rules must be met: 1. <i>hist_max_value</i> = <i>hist_min_value</i> + 2^n , $n \geq 3$ 2. <i>hist_max_value</i> < 2^{29} Units according to the units of the <i>hist_type</i>	RW
18h	5:0	sample_time	Sample time interval. Allowed range for Switch-IB and Spectrum: $\text{time} = 2^{\text{sample\_time}} * 128\text{nSec}$ <i>sample_time</i> range is 0.. <i>cap_max_sample_time</i> -1	RW

Table 1148 - Histograms Hist-type

Entity ID	Description	Entity parameters	Units	Switch-IB	Spectrum
General					
0020h	PortRcvData in the sample time	NA	Bytes	v	v
0021h	PortRcvData_Prio in the sample time	Prio see Table 1149, “PHBR - Hist-Parameters for Prio Layout,” on page 1336	Bytes	v	v
0040h	PortXmitData in the sample time	NA	Bytes	v	v
0041h	PortXmitData_Prio in the sample time	Prio see Table 1149, “PHBR - Hist-Parameters for Prio Layout,” on page 1336	Bytes	v	v
0042h	PortXmitWait	NA	Ticks (<u>Editor note: see PPCNT</u>)	v	v

28.19.53.1 Hist_Parameters for Prio

Table 1149 - PHBR - Hist-Parameters for Prio Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 1150 - PHBR - Hist-Parameters for Prio Fields

Offset	Bits	Name	Description	Access
00h	7:0	prio	Priority Ethernet: Prio (IEEE priority) InfiniBand: VL	RW

28.19.54PHCR - Ports Histogram Configuration Register

The PHCR register allows to configure the parameters for creating an error counter histogram for each of the ports.

The size of the register is flexible and depends on the *num_err_cntrs* field.

Table 1151 - PHCR - Port Histogram Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
opcode								local_port								pnat																				00h
																														num_err_cntrs			04h			
																																				08h
err_cnt_column_range[0](See Table 1153, “PHCR - Port Histogram Counter Configuration Layout,” on page 1338)																															0Ch-48h					

Table 1152 - PHCR - Port Histogram Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	31:29	opcode	Opcode: 0: Query the number of histogram columns. 1: Set the range for each of the columns. 2-7: Reserved	RW
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type: 0: Local_port_number 1: IB_port_number	Index
04h	4:0	num_err_cntrs	Number of counters that are available for counting errors per each of the ports.	RO
0Ch-48h	31:0	err_cnt_column_range[0](See Table 1153, “PHCR - Port Histogram Counter Configuration Layout,” on page 1338)	Configuration per each of the counters See Table 1153, “PHCR - Port Histogram Counter Configuration Layout,” on page 1338	RW

Table 1153 - PHCR - Port Histogram Counter Configuration Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				high_val							low_val				00h	

Table 1154 - PHCR - Port Histogram Counter Configuration Fields

Offset	Bits	Name	Description	Access
00h	11:8	high_val		RW
	3:0	low_val		RW

28.19.55PHRR - Ports Histogram Reading Register

The PHRR allows reading a histogram agent.

Table 1155 - PHRR - Port Histogram Reading Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
clr								local_port								pnat																	00h
																									hist_id								04h
																																08h	
																																0Ch	
bin[0]																																	10-14h
bin[1]																																	18-1Ch
...																																	...
bin[9]																																	58-5Ch

Table 1156 - PHRR - Port Histogram Reading Register Fields

Offset	Bits	Name	Description	Access
00h	31	clr	Clear: 1 - Clear the data in the histogram agent	OP
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type: 0 - Local port number 1 - IB port number	Index
04h	7:0	hist_id	see PHBR	Index
10h-5Ch	64	bin[10]	Value of bin Units according to the units of the <i>hist_type</i> which is used for the <i>hist_id</i> Max bins is according to <i>cap_hist_num_bins</i>	RO

28.19.56PPTT - Port PRBS Transmit Test

The PPTT allows to configure the port transmitter to PRBS test mode.

To activate the TX PRBS mode, the port must be configured to test-mode using PAOS and PPAOS registers.

First PAOS should be set to down, then the prbs parameters are selected using the PPTT and finally the test-mode is enabled by the PPAOS.

While in test-mode, the PTYS protocol/speed selection are ignored.

Table 1157 - PPTT - Port PRBS Transmit Test Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
e			p	c				local_port								pnat			lane				port_type						ls	le	00h					
prbs_modes_cap																																04h				
prbs_mode_admin																												modulation				08h				
lane_rate_cap																																			0Ch	
lane_rate_admin																																				10h
																																14h				
																																18h				

Table 1158 - PPTT - Port PRBS Transmit Test Register Fields

Offset	Bits	Name	Description	Access
00h	31	e	Enable PRBS test mode bit: 0 - PRBS TX is disabled. 1 - PRBS TX is enabled.	RW
	28	p	PRBS TX polarity - NOT gate in PRBS (not Physical lane Polarity) 0 - No polarity inversion. 1 - PRBS TX polarity inversion.	RW
	27	p_c	PRBS TX polarity support indication. if this bit is cleared - bit 28 (<i>'p'</i>) is been ignored	RO
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	lane	Reserved when (<i>le=0</i> or <i>ls = 0</i>) Physical lane number	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tail Other values are reserved. Note: PRBS can be enabled only for two entities that directly connected to each other. For non MCM and no Retimer/Gearbox only 0 allowed. For non MCM and with Retimer/Gearbox 2-1 or 3 or 0. For MCM without Retimer/Gearbox 4-5 or 2 or 0. For MCM with Retimer/Gearbox or 4-5 or 2-1 or 3 or 0. all the port will be in test mode when PRBS is enabled.	Index
	1	ls	Per Lane configuration support/capability: 0 - No support of per lane configuration 1 - Support of per lane configuration	RO
	0	le	Per Lane configuration enable (can be set only if <i>ls = 1</i>): 0 - No per lane configuration 1 - Per lane configurations When <i>le</i> is cleared, <i>lane</i> index is reserved and all PPTT configurations are taking place on all lanes. When <i>le</i> is set, configurations are taking place per lane based on <i>lane</i> index	OP

Table 1158 - PPTT - Port PRBS Transmit Test Register Fields

Offset	Bits	Name	Description	Access
04h	31:0	prbs_modes_cap	PRBS capability (bitmask) Bit 0 - PRBS31 ($x^{31} + x^{28} + 1$) Bit 1 - PRBS23A ($x^{23} + x^{18} + 1$) Bit 2 - PRBS23B ($x^{23} + x^{18} + x^{12} + x^6 + 1$) Bit 3 - PRBS23C ($x^{23} + x^{22} + x^{19} + x^{18} + 1$) Bit 4 - PRBS23D ($x^{23} + x^{19} + x^{18} + x^{11} + 1$) Bit 5 - PRBS7 ($x^7 + x^6 + 1$) Bit 6 - PRBS11 ($x^{11} + x^9 + 1$) Bit 7 - PRBS11A ($x^{11} + x^{10} + x^6 + x^5 + 1$) Bit 8 - PRBS11B ($x^{11} + x^9 + x^6 + x^5 + 1$) Bit 9 - PRBS11C ($x^{11} + x^8 + x^6 + x^4 + 1$) Bit 10 - PRBS11D ($x^{11} + x^7 + x^6 + x^4 + 1$) Bit 11 - PRBS9 ($x^9 + x^5 + 1$) Bit 12 - IDLE using scramble58 ($x^{58} + x^{39} + 1$) Bit 13 - Square_waveA (8 ones , 8 zeros) Bit 14 - Square_waveB (4 ones , 4zeros) Bit 15 - Square_waveC (2 ones , 2zeros) Bit 16 - Square_waveD (one , zero) Bit 17 - PRBS13A ($x^{13} + x^{12} + x^2 + x + 1$) Bit 18 - PRBS13B ($x^{13} + x^7 + x^3 + x^2 + 1$) Bit 19 - PRBS13C ($x^{13} + x^8 + x^4 + x^2 + 1$) Bit 20 - PRBS13D ($x^{13} + x^9 + x^5 + x^2 + 1$) Bit 21- SSPR Bit 22- SSPRQ	RO

Table 1158 - PPTT - Port PRBS Transmit Test Register Fields

Offset	Bits	Name	Description	Access
08h	31:24	prbs_mode_admin	See <i>prbs_mode_cap</i> : 0x0 - PRBS31 ($x^{31} + x^{28} + 1$) 0x1 - PRBS23A ($x^{23} + x^{18} + 1$) 0x2 - PRBS23B ($x^{23} + x^{18} + x^{12} + x^6 + 1$) 0x3 - PRBS23C ($x^{23} + x^{22} + x^{19} + x^{18} + 1$) 0x4 - PRBS23D ($x^{23} + x^{19} + x^{18} + x^{11} + 1$) 0x5 - PRBS7 ($x^7 + x^6 + 1$) 0x6 - PRBS11 ($x^{11} + x^9 + 1$) 0x7 - PRBS11A ($x^{11} + x^{10} + x^6 + x^5 + 1$) 0x8 - PRBS11B ($x^{11} + x^9 + x^6 + x^5 + 1$) 0x9 - PRBS11C ($x^{11} + x^8 + x^6 + x^4 + 1$) 0xA - PRBS11D ($x^{11} + x^7 + x^6 + x^4 + 1$) 0xB - PRBS9 ($x^9 + x^5 + 1$) 0xC - IDLEs using scramble58 ($x^{58} + x^{39} + 1$) 0xD - Square_waveA (8 ones , 8 zeros) 0xE - Square_waveB (4 ones , 4zeros) 0xF - Square_waveC (2 ones , 2zeros) 0x10 - Square_waveD (one , zero) 0x11 - PRBS13A ($x^{13} + x^{12} + x^2 + x + 1$) 0x12 - PRBS13B ($x^{13} + x^7 + x^3 + x^2 + 1$) 0x13 - PRBS13C ($x^{13} + x^8 + x^4 + x^2 + 1$) 0x14 - PRBS13D ($x^{13} + x^9 + x^5 + x^2 + 1$) 0x15 - SSPR 0x16 - SSPRQ	RW
	3:0	modulation	Reserved for speeds below 53.125Gb/s (<i>lane_rate_admin</i> < 13): 0 - NRZ test pattern 1 - PAM4 encoding 2 - PAM4 with precoding	RW
0Ch	31:16	lane_rate_cap	Per lane rate capability (bitmask) Bit 0 - 1GE (1.25 Gb/s) Bit 1 - SDR (2.5 Gb/s) Bit 2 - XAUI/2.5GE (3.125 Gb/s) Bit 3- DDR (5 Gb/s) Bit 4- QDR (10 Gb/s) Bit 5- FDR10 / 10GE/40GE (10.3125 Gb/s) Bit 6- FDR (14.0625 Gb/s) Bit 7- EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) Bit 8 - 50GE-KR4 (12.89 Gb/s) Bit 9 - HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s)	RO

Table 1158 - PPTT - Port PRBS Transmit Test Register Fields

Offset	Bits	Name	Description	Access
10h	31:16	lane_rate_admin	Lane rate be used in PRBS, see <i>lane_rate_cap</i>: 0 - SDR (2.5 Gb/s) 1 - DDR (5 Gb/s) 2 - QDR (10 Gb/s) 3 - FDR10 / 10GE/40GE (10.3125 Gb/s) 4 - FDR (14.0625 Gb/s) 5 - EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) 6 - HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s) 7- 9 - reserved 10 - 1GE (1.25 Gb/s) 11 - XAUI/2.5GE (3.125 Gb/s) 12 - 50GE-KR4 (12.89 Gb/s) Note: All lanes must be set to the same rate	RW

28.19.57PPRT - Port PRBS Receive Test

The PPRT allows to configure the port receiver to perform tuning based on PRBS test mode.

When performing a PRBS based tuning, the port shall be configured to test-mode using PAOS and PPAOS registers.

First PAOS should be sent to down, then the tuning parameters are selected using the PPRT and finally the test-mode is enabled by the PPAOS.

While in PRBS test mode, the PTYS protocols/speeds selection are ignored.

NOTE: Changes in the PPRT (excluding “start tuning”) will take place only after PPAOS will toggle.

Table 1159 - PPRT - Port PRBS Receive Test Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset						
e	s	tun ovr	p	p_c				local_port							pnat			lane				port_type						ls	le	00h								
prbs_modes_cap																															04h							
prbs_mode_admin											tuning_type																	modulation				08h						
lane_rate_cap																																					0Ch	
lane_rate_oper																																						10h
prbs_rx_tuning_status			prbs_lock_status			prbs_lock_status_ext																										14h						
tuning_params																															18h							
																																1Ch						

Table 1160 - PPRT - Port PRBS Receive Test Register Fields

Offset	Bits	Name	Description	Access
00h	31	e	Enable PRBS test mode bit: 0 - PRBS RX is disabled. 1 - PRBS RX is enabled.	RW
	30	s	start tuning: 1 - start RX_tuning based on PRBS pattern Note: assuming peer transmitting PRBS. cannot be set when <i>prbs_rx_tuning_status</i> = 1 (during tuning)	OP
	29	tun_ovr	[Internal]: If set to '1', the value in <i>tuning_type</i> field will override the default adaptation flow (BER Bath for IB and XOR for Eth), which is set automatically when <i>tun_ovr</i> is '0'.	OP
	28	p	PRBS RX polarity - NOT gate in PRBS (not Physical lane Polarity) 0 - No polarity inversion. 1 - PRBS RX polarity inversion.	RW
	27	p_c	PRBS RX polarity support indication if this bit is cleared - bit 28 ('p') is been ignored	RO
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	lane	Reserved when (<i>le</i> =0 or <i>ls</i> = 0) Physical lane number	Index
	7:4	port_type	Supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR main 5 - USR tail Other values are reserved. Note: PRBS can be enabled only for two entities that directly connected to each other. For non MCM and no Retimer/Gearbox only 0 allowed. For non MCM and with Retimer/Gearbox 2-1 or 3 or 0. For MCM without Retimer/Gearbox 4-5 or 2 or 0. For MCM with Retimer/Gearbox or 4-5 or 2-1 or 3 or 0. all the port will be in test mode when PRBS is enabled.	Index
	1	ls	Per Lane configuration support: 0 - No support of per lane configuration 1 - Support of per lane configuration	RO

Table 1160 - PPRT - Port PRBS Receive Test Register Fields

Offset	Bits	Name	Description	Access
00h	0	le	Per Lane configuration enable (can be set only if <i>ls</i> = 1): 0 - No per lane configuration 1 - Per lane configurations When <i>le</i> is cleared, <i>lane</i> index is reserved and all PPRT configurations are taking place on all lanes. When <i>le</i> is set, configurations are taking place per lane based on <i>lane</i> index	OP
04h	31:0	prbs_modes_cap	PRBS capability (bitmask) Bit 0 - PRBS31 ($x^{31} + x^{28} + 1$) Bit 1 - PRBS23A ($x^{23} + x^{18} + 1$) Bit 2 - PRBS23B ($x^{23} + x^{18} + x^{12} + x^6 + 1$) Bit 3 - PRBS23C ($x^{23} + x^{22} + x^{19} + x^{18} + 1$) Bit 4 - PRBS23D ($x^{23} + x^{19} + x^{18} + x^{11} + 1$) Bit 5 - PRBS7 ($x^7 + x^6 + 1$) Bit 6 - PRBS11 ($x^{11} + x^9 + 1$) Bit 7 - PRBS11A ($x^{11} + x^{10} + x^6 + x^5 + 1$) Bit 8 - PRBS11B ($x^{11} + x^9 + x^6 + x^5 + 1$) Bit 9 - PRBS11C ($x^{11} + x^8 + x^6 + x^4 + 1$) Bit 10 - PRBS11D ($x^{11} + x^7 + x^6 + x^4 + 1$) Bit 11 - PRBS9 ($x^9 + x^5 + 1$) Bit 12 - IDLE using scramble58 ($x^{58} + x^{39} + 1$) Bit 13 - Square_wave : Non error based tune Bit 17 - PRBS13A ($x^{13} + x^{12} + x^2 + x + 1$) Bit 18 - PRBS13B ($x^{13} + x^7 + x^3 + x^2 + 1$) Bit 19 - PRBS13C ($x^{13} + x^8 + x^4 + x^2 + 1$) Bit 20 - PRBS13D ($x^{13} + x^9 + x^5 + x^2 + 1$) Bit 21- SSPR Bit 22- SSPRQ	RO
08h	31:24	prbs_mode_admin	See prbs_mode_cap: 0x0 - PRBS31 ($x^{31} + x^{28} + 1$) 0x1 - PRBS23A ($x^{23} + x^{18} + 1$) 0x2 - PRBS23B ($x^{23} + x^{18} + x^{12} + x^6 + 1$) 0x3 - PRBS23C ($x^{23} + x^{22} + x^{19} + x^{18} + 1$) 0x4 - PRBS23D ($x^{23} + x^{19} + x^{18} + x^{11} + 1$) 0x5 - PRBS7 ($x^7 + x^6 + 1$) 0x6 - PRBS11 ($x^{11} + x^9 + 1$) 0x7 - PRBS11A ($x^{11} + x^{10} + x^6 + x^5 + 1$) 0x8 - PRBS11B ($x^{11} + x^9 + x^6 + x^5 + 1$) 0x9 - PRBS11C ($x^{11} + x^8 + x^6 + x^4 + 1$) 0xA - PRBS11D ($x^{11} + x^7 + x^6 + x^4 + 1$) 0xB - PRBS9 ($x^9 + x^5 + 1$) 0xC - IDLEs using scramble58 ($x^{58} + x^{39} + 1$) 0xD - Square_wave : Non error based tune 0x11 - PRBS13A ($x^{13} + x^{12} + x^2 + x + 1$) 0x12 - PRBS13B ($x^{13} + x^7 + x^3 + x^2 + 1$) 0x13 - PRBS13C ($x^{13} + x^8 + x^4 + x^2 + 1$) 0x14 - PRBS13D ($x^{13} + x^9 + x^5 + x^2 + 1$) 0x15 - SSPR 0x16 - SSPRQ	RW

Table 1160 - PPRT - Port PRBS Receive Test Register Fields

Offset	Bits	Name	Description	Access
08h	19:16	tuning_type ¹	15 - Auto (default) 0 - Full adaptation flow BER Bath 1 - Full adaptation flow XOR 2 - Full adaptation flow EO 3 - One grade: BER Bath 4 - One grade: XOR 5 - One grade: EO	RW
08h	3:0	modulation	Reserved for speeds below 53.125Gb/s (<i>lane_rate_admin</i> < 13): 0 - NRZ test pattern 1 - PAM4 encoding 2 - PAM4 with precoding 3 - PAM4 without gray no precoding	RW
0Ch	31:16	lane_rate_cap	Per lane rate capability (bitmask) Bit 0 - 1GE (1.25 Gb/s) Bit 1 - SDR (2.5 Gb/s) Bit 2 - XAUI/2.5GE (3.125 Gb/s) Bit 3- DDR (5 Gb/s) Bit 4- QDR (10 Gb/s) Bit 5- FDR10 / 10GE/40GE (10.3125 Gb/s) Bit 6- FDR (14.0625 Gb/s) Bit 7- EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) Bit 8 - 50GE-KR4 (12.89 Gb/s) Bit 9 - HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s)	RO
10h	31:16	lane_rate_oper	Lane rate be used in PRBS, see lane_rate_cap: 0 - SDR (2.5 Gb/s) 1 - DDR (5 Gb/s) 2 - QDR (10 Gb/s) 3 - FDR10 / 10GE/4 0GE (10.3125 Gb/s) 4 - FDR (14.0625 Gb/s) 5 - EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) 6 - HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s) 7 - 9 - reserved 10 - 1GE (1.25 Gb/s) 11 - XAUI/2.5GE (3.125 Gb/s) 12 - 50GE-KR4 (12.89 Gb/s) Note: All lanes must be set to the same rate	RW

Table 1160 - PPRT - Port PRBS Receive Test Register Fields

Offset	Bits	Name	Description	Access
14h	31:28	prbs_rx_tuning_status	Tuning status: 0 - PRBS mode tuning was not performed. 1 - Performing PRBS mode tuning. 2 - PRBS mode tuning completed. Note: Once exiting prbs test mode the <i>prbs_rx_tuning_status</i> will return to 0.	RO
	27:24	prbs_lock_status	PRBS lock on PRBS pattern status for lanes 3-0: 0 - Not locked 1 - Locked Note: Once exiting prbs test mode the <i>prbs_rx_tuning_status</i> will return to 0 (Not locked)	RO
	23:20	prbs_lock_status_ext	PRBS lock on PRBS pattern status for lanes 7-4: 0 - Not locked 1 - Locked Note: Once exiting prbs test mode the <i>prbs_rx_tuning_status</i> will return to 0 (Not locked)	RO
18h	31:0	tuning_params	Valid only when <i>tuning_type</i> is “EO”. Reserved for all other type. See	RO

1. When square_wave is set, BER bath can't be supported

Table 1161 - Tuning_params Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
																									tuning_time											00h

Table 1162 - Tuning_params Fields

Offset	Bits	Name	Description	Access
00h	7:0	tuning_time		RW

28.19.58[Internal] PMPT - Port Module PRBS Test

The PMPT allows to configure the module to PRBS test mode.

To activate the module to PRBS mode, the module must be in active status in PMAOS.

Table 1163 - PMPT - Port Module PRBS Test Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																				
e				ls				le				swap_cap				invt_cap				module								host_media				lane								status								00h				
prbs_modes_cap																																04h																				
prbs_mode_admin																								swap_admin								invt_admin				modulation								08h								
lane_rate_cap																																																0Ch				
lane_rate_admin																																																10h				
																																																14h				
																																																				18h

Table 1164 - PMPT - Port Module PRBS Test Register Fields

Offset	Bits	Name	Description	Access
00h	31	e	Enable PRBS test mode bit: 0 - PRBS is disabled. 1 - PRBS is enabled.	RW
	30	ls	Per Lane configuration support/capability: 0 - No support of per lane configuration 1 - Support of per lane configuration	RO
	29	le	Per Lane configuration enable (can be set only if <i>ls</i> = 1): 0 - No per lane configuration 1 - Per lane configurations When <i>le</i> is cleared, <i>lane</i> index is reserved and all PMPT configurations are taking place on all lanes. When <i>le</i> is set, configurations are taking place per lane based on <i>lane</i> index	OP
	28	swap_cap	PAM4 MSB<-> LSB swapping is supported by the module	RO
	27	inv_t_cap	PRBS inversion is supported	RO
	23:16	module		Index
	12	host_media	Host or Media lanes	Index
	11:8	lane	Reserved when (<i>le</i> =0 or <i>ls</i> = 0) Physical lane number	Index
	3:0	status	0 - Normal mission mode. 1 - Module is disabled/not connected. 2 - PRBS process 3 - unsupported configuration setting 4 - PRBS traffic both ingress and egress with prbs checker 5 - PRBS traffic without checker 6 - PRBS checker only	RO
04h	31:0	prbs_modes_cap	Supported module's PRBS capability (bitmask) Bit 0 - PRBS31 ($x^{31} + x^{28} + 1$) Bit 1 - PRBS23 ($x^{23} + x^{18} + 1$) Bit 5 - PRBS7 ($x^7 + x^6 + 1$) Bit 6 - PRBS11 ($x^{11} + x^9 + 1$) Bit 11 - PRBS9 ($x^9 + x^5 + 1$) Bit 17 - PRBS13 ($x^{13} + x^{12} + x^2 + x + 1$) Bit 21 - SSPR Bit 22 - SSPRQ	RO

Table 1164 - PMPT - Port Module PRBS Test Register Fields

Offset	Bits	Name	Description	Access
08h	31:24	prbs_mode_admin	See prbs_mode_cap : 0x0 - PRBS31 ($x^{31} + x^{28} + 1$) 0x1 - PRBS23 ($x^{23} + x^{18} + 1$) 0x5 - PRBS7 ($x^7 + x^6 + 1$) 0x6 - PRBS11 ($x^{11} + x^9 + 1$) 0xB - PRBS9 ($x^9 + x^5 + 1$) 0x11 - PRBS13 ($x^{13} + x^{12} + x^2 + x + 1$) 0x15 - SSPR 0x16 - SSPRQ	RW
	9	swap_admin	PAM4 MSB<-> LSB swapping enable bit, see swap_cap. 0 - NO MSB<-> LSB swapping 1 - MSB <-> LSB swapping	RW
	8	invnt_admin	PRBS inversion enable bit, see ivnt_cap: 0 - NO PRBS inversion 1 - PRBS inversion	RW
	3:0	modulation	Reserved for speeds below 53.125Gb/s (lane_rate_admin < 7): 0 - NRZ test pattern 1 - PAM4 encoding 2 - PAM4 with precoding	RW
0Ch	31:16	lane_rate_cap	Per lane rate capability (bitmask) Bit 0 - 1GE (1.25 Gb/s) Bit 1 - SDR (2.5 Gb/s) Bit 2 - Reserved [Internal] 2.5GE (3.125 Gb/s) Bit 3- 10GE/40GE (10.3125 Gb/s) Bit 4- FDR (14.0625 Gb/s) Bit 5- EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) Bit 6- Reserved [Internal] FDR (14.0625 Gb/s) Bit 7- HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s)	RO
10h	31:16	lane_rate_admin	Lane rate be used in PRBS, see lane_rate_cap : 0 - Non selected. 1 - 1GE (1.25 Gb/s) 2 - SDR (2.5 Gb/s) 8 - 10GE/40GE (10.3125 Gb/s) 16 - FDR (14.0625 Gb/s) 32 - EDR / 25GE / 50GE / 100GE (25.78125 Gb/s) 128 - HDR /50GE / 100GE / 200GE / 400GE (26.5625Gbd / 53.125Gb/s) TBD Note: All lanes must be set to the same rate	RW

28.19.59[Internal] PMPD - Port Module PRBS Diagnostics

The PMPD shows SI and statical information on module's PRBS.

To activate the module to PRBS mode, see PMPT.

Table 1165 - PMPD - Port Module PRBS Diagnostics Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								module											host media	lane									status				00h
prbs_bits_high																															04h		
prbs_bits_low																															08h		
prbs_errors_high																															0Ch		
prbs_errors_low																															10h		
measured_snr															peak_detect																14h		
																ber_magnitude									ber_coef				18h				
																																1Ch-2Ch	

Table 1166 - PMPD - Port Module PRBS Diagnostics Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module		Index
	12	host_media	Module lane number	Index
	11:8	lane	Module lane number	Index
	3:0	status	0 - Normal mission mode. 1 - PRBS checker is not locked. 2 - PRBS checker is locked.	RO
04h	31:0	prbs_bits_high	This counter provides information on the total amount of prbs traffic (bits) received on module's side.	RO
08h	31:0	prbs_bits_low		
0Ch	31:0	prbs_errors_high	This counter provides information on the total amount of prbs errors detected on module's side.	RO
10h	31:0	prbs_errors_low		
14h	31:16	measured_snr	SNR in dB 0 - not supported.	RO
	15:0	peak_detect		RO

Table 1166 - PMPD - Port Module PRBS Diagnostics Register Fields

Offset	Bits	Name	Description	Access
18h	15:8	ber_magnitude	BER = raw_ber_coef*10^(-raw_ber_magnitude)	RO
	3:0	ber_coef		RO

28.19.60PCMR - Ports Check Mask Register

The PCMR register configures and reports the port Ethernet checking list.

For NIC devices supported only when *HCA_CAP.port_checks=1*

Table 1167 - PCMR - Ports Check Mask Register Layout

Offset	00h	04h	08h
0		pes cap	pes chk
1		fcs cap	fcs chk
2		sht cap	sht chk
3		mtu cap	mtu chk
4		type cap	type chk
5		len cap	len chk
6			
7		fc cap	fc chk
8		cp cap	cp chk
9			
10			
11			
12			
13		rx ts over crc cap	rx ts over crc
14		rx fcs drop cap	rx fcs drop
15		tx fcs recalc cap	tx fcs recalc
16	local_port	tx ts over crc cap	tx ts over crc
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29		entropy gre calc cap	entropy gre calc
30		entropy calc cap	entropy calc
31		force entropy cap	force entropy

Table 1168 - PCMR - Ports Check Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number [internal] NOTE: no support of CPU port, see footnote for implications	Index

Table 1168 - PCMR - Ports Check Mask Register Fields

Offset	Bits	Name	Description	Access
04h	31	force_entropy_cap	Specifies the egress tunnel protocol entropy calculation force capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	30	entropy_calc_cap	Specifies the egress tunnel protocol entropy calculation enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	29	entropy_gre_calc_cap	Specifies the egress GRE tunnel protocol entropy calculation enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	16	tx_ts_over_crc_cap	Specifies the egress timestamps enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RW
	15	tx_fcs_recalc_cap	Specifies the egress drop enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	14	rx_fcs_drop_cap	Specifies the ingress drop enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	13	rx_ts_over_crc_cap	Specifies the ingress timestamps enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability:	RO
	8	cp_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>Control packet check disable/enable capability</i>	RO
	7	fc_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>flow-control packet check disable/enable capability</i>	RO
	5	len_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>LENGTH MISMATCH check disable/enable capability (Length in LengthType is does not match actual payload length)</i>	RO
	4	type_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>FRAME TYPE check disable/enable capability (out-of-range Type-Length field)</i>	RO
	3	mtu_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>MTU error - check disable/enable capability (data packet's actual length over mtu_frame_size)</i>	RO
	2	sht_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: <i>SHORT FRAME check disable/enable capability</i>	RO

Table 1168 - PCMR - Ports Check Mask Register Fields

Offset	Bits	Name	Description	Access
04h	1	fcs_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: FCS check disable/enable capability	RO
	0	pcs_cap	Specifies the check enable/disable capabilities of the local port. A bit set to 1 for affirmation of supported enable/disable capability: PHY/PCS check disable/enable capability	RO
08h	31	force_entropy	0x0: DEFAULT - use device defaults for tunnel entropy calculation. This mode should be used if entropy calculation requirement is unknown. 0x1: FORCE - force the entropy calculation according to <i>entropy_calc</i> and <i>entropy_gre_calc</i> .	RW
	30	entropy_calc	0x0: DISABLE - entropy calculation is disabled for all supported tunneling protocols. 0x1: ENABLE - entropy calculation is enabled for all supported tunneling protocols. GRE also depends on the <i>entropy_gre_calc</i> valid only if <i>force_entropy</i> is activated.	RW
	29	entropy_gre_calc	0x0: DISABLE - entropy calculation is disabled for GRE tunneling protocol. 0x1: ENABLE - entropy calculation is enabled for GRE tunneling protocol, if <i>entropy_calc</i> is enabled as well. valid only if <i>force_entropy</i> is activated.	RW
	16	tx_ts_over_crc	TX TimeStamp over FCS 0: no TS over FCS 1: TS over FCS Reserved when <i>tx_fcs_recalc</i> is set	RW
	15	tx_fcs_recalc ¹	TX FCS drop control: 0: egress port will recalculate and overwrite FCS value (Default) [Internal] Egress port will drop packets with bad FCS on payload. 1: egress port will not overwrite FCS value [Internal] Egress port will not drop packets with bad FCS on payload. NOTE: when this bit is set, any modification of the packet will result in a bad FCS.	RW
	14	rx_fcs_drop	RX FCS drop control: 0: ingress port will drop a packet with bad FCS (Default) 1: ingress port will not drop a packet with bad FCS NOTE: All FCS counters (PPCNT) are valid for both drop modes. NOTE: Reserved when <i>PCMR.fcs_cap=1</i> & <i>PCMR.fcs_chk=0</i>	RW
	13	rx_ts_over_crc	RX TimeStamp over FCS 0: no TS over FCS 1: TS over FCS	RW

Table 1168 - PCMR - Ports Check Mask Register Fields

Offset	Bits	Name	Description	Access
08h	8	cp_chk	Control packet check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
	7	fc_chk	flow-control packet check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
	5	len_chk	LENGTH MISMATCH check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
	4	type_chk	FRAME TYPE check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
	3	mtu_chk	MTU check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
	2	sht_chk	SHORT FRAME check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
08h	1	fcs_chk	FCS check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW
08h	0	pcs_chk	PHY/PCS error check Values: 0: disable check. 1: perform check Note: when disabling a check, it will also affects the relevant counters.	RW

1. Note: Packets must path through ingress and egress ports with the same configuration.

Ingress port: PCMR.rx_fcs_drop=0 , Egress port: PCMR.tx_fcs_drop=0 – good, regular, default mode

Ingress port: PCMR.rx_fcs_drop=1 , Egress port: PCMR.tx_fcs_drop=1 – good, fcs-accept mode

Ingress port: PCMR.rx_fcs_drop=0 , Egress port: PCMR.tx_fcs_drop=1 – misconfiguration, all egressed packets will have corrupt fcs

Ingress port: PCMR.rx_fcs_drop=1 , Egress port: PCMR.tx_fcs_drop=0 – misconfiguration, all egressed packets will be discarded

28.19.61PDDR - Port Diagnostics Database Register

This register enables to read the Phy debug database

Table 1169 - PDDR - Port Diagnostics Database Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-FCh

Table 1170 - PDDR - Port Diagnostics Database Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number.	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0: Local_port_number 1: IB_or_label_port_number 3: Out_of_band_or_PCI - [Internal]	Index
04h	31	clr	Clear page, based on internal page clear bit mask. This bit can be set for both Set() and Get() operation.	OP
04h	7:0	page_select	page select index: 0: Operational_info_page 1: Troubleshooting_info_page 2: Phy_info_page - [Internal] 3: Module_info_page 4: port_events_page - [internal] 5: device_events_page - [internal] 6: link_down_info - [EMC only] 7: link_down_events - [internal] 8: Link_up_info - [Internal]	Index
08h-FCh	1984	page_data	See Table 1171, "Operation Info Page Layout," on page 1363 See Table 1189, "Phy info page layout," on page 1370 See Table 1197, "Troubleshooting info Page layout," on page 1390 See Table 1195, "Module Info Page Layout," on page 1380 See Table 1205, "Device events Page Layout," on page 1399 See Table 1203, "Port events Page Layout," on page 1396 See Table 1191, "Link down info page layout," on page 1377 See Table 1193, "Link down events page layout," on page 1379 See Table 1207, "Link Up Info Page Layout," on page 1402	RO

Table 1171 - Operation Info Page Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset						
								proto_active				neg_-mode_active																					00h					
phy_mngr_fsm_state								eth_an_fsm_state								ib_phy_fsm_state																						04h
phy_manager_link_enabled																																08h						
core_to_phy_link_enabled																																0Ch						
cable_proto_cap																																10h						
link_active																																14h						
retran_mode_active								retran_mode_request																						loopback_-mode				18h				
fec_mode_active																fec_mode_request																1Ch						
profile_fec_in_use																									eth_25g_50g_fec_support				eth_100g_fec_support				20h					
																																	24h-F4h					

Table 1172 - Operation Info Page Fields

Offset	Bits	Name	Description	Access
00h	23:20	proto_active	Active protocol: Bit 0: InfiniBand Bit 2: Ethernet	RO
00h	19:16	neg_mode_active	0 - protocol was not negotiated (force mode) 1 - MLPN rev 0 negotiated 2 - CL73 Ethernet negotiated 3 - Protocol according to Parallel detect (remote port in force mode) 4 - Standard IB negotiated 5 - MLPN rev 1 negotiated 6 - High Speed tuning negotiated (MLPN Rev 1).	RO

Table 1172 - Operation Info Page Fields

Offset	Bits	Name	Description	Access
04h	31:24	phy_mn-gr_fsm_state	FW Phy Manager FSM state: 0 - Disabled 1 - Open port 2 - Polling 3 - Active / Linkup 4 - Close port	RO
	23:16	eth_an_fsm_state	Ethernet (CL73) Auto-negotiation FSM state: 0x0 - ETH_AN_FSM_ENABLE 0x1 - ETH_AN_FSM_XMIT_DISABLE 0x2 - ETH_AN_FSM_ABILITY_DETECT 0x3 - ETH_AN_FSM_ACK_DETECT 0x4 - ETH_AN_FSM_COMPLETE_ACK 0x5 - ETH_AN_FSM_AN_GOOD_CHECK 0x6 - ETH_AN_FSM_AN_GOOD 0x7 - ETH_AN_FSM_NEXT_PAGE_WAIT [Internal] 0x8 - ETH_AN_FSM_LINK_STAT_CHECK [Internal] 0x9 - ETH_AN_FSM_EXTRA_TUNE [Internal] 0xa - ETH_AN_FSM_FIX_REVERSALS [Internal] 0xb - ETH_AN_FSM_IB_FAIL [Internal] NOTE: Check if we need post_lock_tune?	
	15:8	ib_phy_fsm_state	FW IB state machine: 0x0 - IB_AN_FSM_DISABLED 0x1 - IB_AN_FSM_INITIALY 0x2 - IB_AN_FSM_RCVR_CFG 0x3 - IB_AN_FSM_CFG_TEST 0x4 - IB_AN_FSM_WAIT_RMT_TEST 0x5 - IB_AN_FSM_WAIT_CFG_ENHANCED 0x6 - IB_AN_FSM_CFG_IDLE 0x7 - IB_AN_FSM_LINK_UP [Internal] NOTE: Do we need "module test" state	RO
08h	31:0	phy_manager_link_enabled	For IB: See Table 1173, "PDDR - Phy manager link enabled IB layout," on page 1367 For Ethernet: See Table 1175, "PDDR - Phy manager link enabled Eth layout," on page 1367	RO
0Ch	31:0	core_to_phy_link_enabled	For IB: See Table 1177, "PDDR - core2phy link enabled IB layout," on page 1368 For Ethernet: See Table 1179, "PDDR - Core2Phy link enabled Eth layout," on page 1368	RO

Table 1172 - Operation Info Page Fields

Offset	Bits	Name	Description	Access
10h	31:0	cable_proto_cap	For IB: See Table 1181, “PDDR - cable cap IB layout,” on page 1369 For Ethernet: See Table 1183, “PDDR - cable cap Eth layout,” on page 1369	RO
14h	31:0	link_active	For IB: See Table 1185, “PDDR - Link Active IB layout,” on page 1370 For Ethernet: See Table 1187, “PDDR - Link Active Eth layout,” on page 1370	RO
18h	31:24	retran_mode_active	Retransmission active 0 - No retransmission 1 - LLR ~~2 - PLR~~ for supported retransmission mode see <i>PPLM.retransmission_capability</i>	RO
	23:16	retran_mode_request	Retransmission mode requested 0 - No retransmission 1 - LLR ~~2 - PLR~~ for supported retransmission mode see <i>PPLM.retransmission_capability</i>	RO
18h	3:0	loopback_mode	0 - Phy remote loopback 2 - Phy local loopback. When set the port’s egress traffic is looped back to the receiver and the port transmitter is disabled. 4 - External local loopback. Enables the port’s transmitter to link with the port’s receiver using an external loopback connector. Bits 3-7 are reserved.	RO
1Ch	31:16	fec_mode_active	FEC mode active 0 - No FEC 1 - Firecode FEC 2 - Standard RS-FEC - RS(528,514) 3 - Standard LL RS-FEC - RS(271,257) [Internal] 4 - Mellanox Strong RS-FEC - RS(277,257) [Internal] 5 - Mellanox LL RS-FEC - RS(163,155) ~~[Internal] 6 - Zero Latency FEC (ZLF)~~ [Internal] 6 - Reserved Bit 7 - Standard RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC [Internal] Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR for supported retransmission mode see <i>PPLM.fec_mode_capability</i>	RO
	15:0	fec_mode_request	FEC mode request See fec_mode_active for supported retransmission mode see <i>PPLM.fec_mode_capability</i>	RO

Table 1172 - Operation Info Page Fields

Offset	Bits	Name	Description	Access
20h	31:16	pro-file_fec_in_use	The profile that has been selected: Bit 0 - IB spec / legacy (See profiles description) Bit 1 - Internal ports (Backplane) Bit 2 - Passive copper - Short Bit 3 - Passive copper - Medium Bit 4 - Passive copper - Long Bit 5 - Active optics / copper short reach (<20m) Bit 6 - Optics long reach (>20m) Bit 7 - NO-FEC Bit 8 - FEC-ON	RO
20h	7:4	eth_25g_50g_fec_support	FEC 25G/50G (25Gb/s per lane) supported FEC include override masking , should reflect current phy configuration Bit 0 - No FEC Bit 1 - Firecode FEC Bit 2 - Standard RS-FEC - RS(528,514) ~~for supported retransmission mode see PPLM.<TBD>_capability~~	RO
	3:0	eth_100g_fec_support	FEC 100G (25Gb/s per lane) supported FEC include override masking , should reflect current phy configuration Bit 0 - No FEC Bit 2 - Standard RS-FEC - RS(528,514) ~~for supported retransmission mode see PPLM.<TBD>_capability~~	RO

28.19.61.1 phy_manager_link_enabled IB

Table 1173 - PDDR - Phy manager link enabled IB layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
phy_manager_link_width_enabled																phy_manager_link_proto_enabled																00h

Table 1174 - PDDR - Phy manager link enabled IB Fields

Offset	Bits	Name	Description	Access
00h	31:16	phy_manager_link_width_enabled	ib link enabled width: Bit 0 - 1x Bit 1 - 2x Bit 2 - 4x Bit 3 - 8x Other - reserved	RO
	15:0	phy_manager_link_proto_enabled	ib link enabled speed: Bit 0 - SDR Bit 1 - DDR Bit 2 - QDR Bit 3 - FDR10 Bit 4 - FDR Bit 5 - EDR Bit 6 - HDR Bit 7 - NDR	RO

28.19.61.2 phy_manager_link_enabled Ethernet

Table 1175 - PDDR - Phy manager link enabled Eth layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
phy_manager_link_eth_enabled																																00h

Table 1176 - PDDR - Phy manager link enabled Eth Fields

Offset	Bits	Name	Description	Access
00h	31:0	phy_manager_link_eth_enabled	Ethernet protocols admin state: see PTYS.ext_eth_proto_admin	RO

28.19.61.3 core_to_phy_link_enabled IB

Table 1177 - PDDR - core2phy link enabled IB layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
core_to_phy_link_width_enabled																core_to_phy_link_proto_enabled																00h

Table 1178 - PDDR - core2phy link enabled IB Fields

Offset	Bits	Name	Description	Access
00h	31:16	core_to_phy_link_width_enabled	ib link enabled width: Bit 0: 1x Bit 1: 2x Bit 2: 4x Bit 3: 8x Other - reserved	RO
	15:0	core_to_phy_link_proto_enabled	ib link enabled speed: Bit 0: SDR Bit 1: DDR Bit 2: QDR Bit 3: FDR10 Bit 4: FDR Bit 5: EDR Bit 6: HDR Bit 7: NDR	RO

28.19.61.4 core_to_phy_link_enabled Ethernet

Table 1179 - PDDR - Core2Phy link enabled Eth layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
core_to_phy_link_eth_enabled																																00h

Table 1180 - PDDR - Core2Phy link enabled Eth Fields

Offset	Bits	Name	Description	Access
0h	31:0	core_to_phy_link_eth_enabled	Ethernet protocols admin state: see PTYS.ext_eth_proto_admin	RO

28.19.61.5 cable capability IB

Table 1181 - PDDR - cable cap IB layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
cable_link_width_cap																cable_link_speed_cap																00h

Table 1182 - PDDR - PDDR - cable cap IB Fields

Offset	Bits	Name	Description	Access
00h	31:16	cable_link_widt h_cap	Cable support IB width: Bit 0: 1x Bit 1: 2x Bit 2: 4x Bit 3: 8x Other - reserved	RO
	15:0	cable_link_- speed_cap	Cable support IB speed: Bit 0: SDR Bit 1: DDR Bit 2: QDR Bit 3: FDR10 Bit 4: FDR Bit 5: EDR Bit 6: HDR Bit 7: NDR	RO

28.19.61.6 cable capability Ethernet

Table 1183 - PDDR - cable cap Eth layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
cable_ext_eth_proto_cap																																00h

Table 1184 - PDDR - cable cap Eth Fields

Offset	Bits	Name	Description	Access
00h	31:0	cable_ex- t_eth_proto_cap	Cable Ethernet protocols cap. For opcode definition see: PTYS.ext_eth_proto_cap	RO

28.19.61.7 Link Active IB

Table 1185 - PDDR - Link Active IB layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
link_width_active																link_speed_active																00h

Table 1186 - PDDR - Link Active IB Fields

Offset	Bits	Name	Description	Access
00h	31:16	link_width_active	ib link active width: Bit 0: 1x Bit 1: 2x Bit 2: 4x Bit 3: 8x Other - reserved	RO
	15:0	link_speed_active	ib link active speed: Bit 0: SDR Bit 1: DDR Bit 2: QDR Bit 3: FDR10 Bit 4: FDR Bit 5: EDR Bit 6: HDR Bit 7: NDR	RO

28.19.61.8 Link Active Ethernet

Table 1187 - PDDR - Link Active Eth layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
link_eth_active																																00h

Table 1188 - PDDR - Link Active Eth Fields

Offset	Bits	Name	Description	Access
00h	31:0	link_eth_active	Ethernet protocols active : see PTYS.ext_eth_proto_oper	RO

Table 1189 - Phy info page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
remote_device_type																port_notifications																00h

Table 1189 - Phy info page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																78h-F4h

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
00h	31:24	remote_device_type	Remote device type: 0 - Unknown (3rd party , Force , legacy MLPN) 1 - CX4 2 - CX4_LX 3 - CX5 4 - CX5_LX 5 - CX6 6 - CX6_LX 7-99 Reserved 100 - SwitchIB 101 - Spectrum 102 - SwitchIB-2 103 - SwitchIB-3 104 - Spectrum-2 105-255 Reserved	RO
	7:0	port_notifications	Port notifications bit mask Bit 0 - Link partner advertised capabilities Bit 1 - Speed degradation	RO
04h	31:16	num_of_negotiation_attempts	Negotiation attempt counter	RO
	15:8	ib_revision	IB TS revision	RO
	7:0	lp_ib_revision	Link partner IB TS revision	RO

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
08h	31:24	hw_link_ - phy_state	IBP HW FSM Reflected State: 0x10 - sleeping delay 0x11 - sleeping quiet 0x20 - polling active 0x21 - polling quiet 0x30 - disable 0x40 - config debounce 0x41 - config receiver 0x42 - config wait remote 0x43 - config tx reverse lanes 0x44 - config enhanced 0x45 - config test 0x46 - config wait remote test 0x47 - config wait cfg enhanced 0x48 - config idle 0x50 - linkup 0x60 - recover retrain 0x61 - recover wait remote 0x62 - recover idle 0x70 - test Force Modes: 0x80 - Force send ts1 (command may be given only on disable state) 0x90 - Force send ts2 0xA0 - Force Sent Idle 0xB0 - Force send ts Mlnx 0xC0 - Force send ts3 0xD0 - Force LinkUp	RO
	15:0	phy_manger_dis- able_mask	Phy manager disable bit mask: Bit 0 - Module not present (module absence/cage power off) Bit 1 - PAOS command Bit 2 - MAD IB PortInfo down command Bit 3 - Long range cable (Non Mellanox) Bit 4 - Verification command Bit 5 - ekey command Bit 6 - High power - cable require higher power than allowed. Bit 7 - Unknown Identifier (Module) Bit 8 - PAOS up once Bit 9 - Stamping failure Bit 10 - Calibration not done [Internal] Bit 11 - Module Locking [Internal] Bit 13 - Power budget exceeded [Internal] Bit 14 - Interrupt Event / Module Temp [Internal] Bit 15 - TEC flow / module bring up issue	RO

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	pcs_phy_state	PCS HW state: Bits 0-3: Block lock Bits 4-7: Reserved Bits 8-11: AM lock Bits 12-15: Reserved Bits 16-19: FEC lock Bits 20-23: Reserved Bits 24: Align_status Bits 25: Hi_BER	RO
10h	31:0	lp_proto_enabled	IB ports : Link partner advertised speeds (first TS3) See ib_link_speed_enabled encoding. Ethernet ports: Link partner advertised Ethernet protocols active state: see PTYS.lp_advertise	RO
14h	31:16	lp_fec_mode_support	Reserved when negotiation wasn't performed according to port_notifications.Link partner advertised capabilities value. Advertised link partner FEC mode support Bit 0 - No FEC Bit 1 - Firecode FEC Bit 2 - Standard RS-FEC - RS(528,514) Bit 3 - Standard LL RS-FEC - RS(271,257) Bit 4 - Mellanox Strong RS-FEC - RS(277,257) Bit 5 - Mellanox LL RS-FEC - RS(163,155) ~~Bit 6 - Zero Latency FEC (ZLF)~~ Bit 6 - Reserved Bit 7 - Standard RS-FEC (544,514) [Internal] Bit 8 - Zero Latency FEC [Internal] Bit 12 - RS-FEC (544,514) + PLR [Internal] Bit 13 - LL-FEC (271,257) + PLR for supported retransmission mode see PPLM.fec_mode_capability	RO
	15:0	lp_fec_mode_request	Reserved when negotiation wasn't performed according to port_notifications.Link partner advertised capabilities value. Advertised link partner FEC mode request	RO

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
18h	31:0	ib_last_link_down_reason	Bit 0 - heartbeat_ack_received Bit 1 - heartbeat_send_received Bit 2 - heartbeat_err Bit 3 - tx_width_reduction_done_1x Bit 4 - tx_width_reduction_done_full Bit 5 - rx_width_reduction_done_1x Bit 6 - rx_width_reduction_done_full Bit 7 - width_reduction_timeout Bit 8 - ibl_link_retrain Bit 9 - rx_comskp_timeout Bit 10 - fifo_full_err Bit 11 - ts_at_linkup Bit 12 - minor_threshold_reached Bit 13 - link_fail Bit 14 - rx_eb_full_r Bit 15 - rx_8_10_lane_err Bit 16 - llr_link_retrain_set Bit 17 - fc_timeout Bit 18 - phy_error Bit 19 - lli_err Bit 20 - excessive_buffer_err Bit 21 - remote_init Bits 31-22 - Reserved	RO
1Ch	7:0	eth_last_link_down_lane[4]	Bit 0 - block_lock_failed Bit 1 - skip_detected Bit 2 - fec_sync_failed Bit 3 - fec_block_sync Bit 4 - fec_block_sync_lost	RO
24h	31:0	speed_deg_db	Bit 0 - Speed degradation database Valid Bit 1 - Speed degradation Serdes Rx database valid Bits 3-2 - reserved Bit 4 - rx_reversal Bit 5 - tx_reversal Bit 7-6 - reserved Bits 11:8 - failed_qdr/fdr10 - bit per lane. Bits 15:12 - failed_fdr - bit per lane. Bits 19:16 - failed_edr - bit per lane. Bits 24:20 - peer_failed_test Bits 26-25 - reserved Bit 27 - first_test_speed	RO

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
28h	23:0	degrade_grade_lane0	Valid only when <i>port_notifications.Speed degradation</i> is asserted	RO
2Ch	23:0	degrade_grade_lane1		RO
30h	23:0	degrade_grade_lane2		RO
34h	23:0	degrade_grade_lane3		RO
48h	7:0	kr_start-up_fsm_lane[4]	Per lane KR startup state: 0 - KR_FSM_INITIALIZE 1 - KR_FSM_SEND_TRAINING 2 - KR_FSM_TRAIN_LOCAL_TX 3 - KR_FSM_TRAIN_LOCAL_RX 4 - KR_FSM_TRAIN_REMOTE 5 - KR_FSM_LINK_READY 6 - KR_FSM_SEND_DATA	RO
50h	31:0	eth_an_debug_indication	eth_an_debug bit mask: Bit 0 - Force link up Bit 1 - No HCD Bit 2 - Entered ACK_detect Bit 3 - Entered Good Bit 4 - Entered Good_Check Bit 5 - Entered Extra_tune Bit 6 - Entered Fix_Reversals Bit 7 - Entered Next_Page Bit 8 - Entered Sub-FSM Fail Bit 9 - Tuning timeout Bit 10 - No markers detected (during Good check) Bit 11 - Do KR-startup Bits 15:18 - KR startup failure mask	RO
54h	15:0	ib_phy_fsm_state_trace	FW IB state machine: Bit 0 - entered IB_AN_FSM_DISABLED Bit 1 - entered IB_AN_FSM_INITIALY Bit 2 - entered IB_AN_FSM_RCVR_CFG Bit 3 - entered IB_AN_FSM_CFG_TEST Bit 4 - entered IB_AN_FSM_WAIT_RMT_TEST Bit 5 - entered IB_AN_FSM_WAIT_CFG_ENHANCED Bit 6 - entered IB_AN_FSM_CFG_IDLE Bit 7 - entered IB_AN_FSM_LINK_UP Bit 8 - Failed from CFG_IDLE	RO

Table 1190 - Phy info page Fields

Offset	Bits	Name	Description	Access
58h-5Ch	31:16	kr_startup_debug_indications[4]	kr_stratup_debug_indication_<i> bit mask: Bit 0 - PRBS lock failed Bit 1 - Entered send training Bit 2 - Entered train local RX Bit 3 - Entered train remote Bit 4 - Entered link ready Bit 5 - Entered Send_Data Bit 6 - Didn't get receiver ready Bit 7 - tuning timeout Bit 8 - No frame lock Bit 9 - No response from peer Bit 10 - frames lock tuning not done	RO
68h	3:0	irisc_status		RO
6Ch	31:0	stamping_reason	Bit 0 - com_codes_is_zero Bit 1- rx_cdr_check_force_mode Bit 2- com_code_complainece Bit 3- eth_56g_stamped Bit 4- non_mlx_qsfp_tranciver Bit 5- "non_mlx_sfp_tranciver" Bit 6-ib_comp_codes Bit 7-edr_comp Bit 8-fdr_comp	RO
70h	31:0	kr - frame_lock_tuning_failure_events_count	Number of times that fast tuning (50ms) for KR lock failed to achieve frame lock.	RO
74h	31:0	kr_full_tuning_failure_count	Number of times that full tuning (0.5sec/3sec) for KR full tuning flow failed to achieve desire SI performance	RO

Table 1191 - Link down info page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																down_blame	00h
																															local_reason_opcode	04h	
																															remote_reason_opcode	08h	
																															e2e_reason_opcode	0Ch	
				last_ber_mantissa				last_ber_exp												ber_mantissa				ber_exp				10h					

Table 1191 - Link down info page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																14h
																																18h
																																1Ch-2Ch

Table 1192 - Link down info page Fields

Offset	Bits	Name	Description	Access
00h	3:0	down_blame	Which receiver caused last link down: 0: Unknown 1: Local_phy 2: Remote_phy	RO
04h	7:0	local_reason_op-code	Opcode of link down reason: 0: No_link_down_indication 1: Unknown_reason 2: Hi_SER_or_Hi_BER 3: Block_Lock_loss 4: Alignment_loss 5: FEC_sync_loss 6: PLL_lock_loss 7: FIFO_overflow 8: false_SKIP_condition 9: Minor_Error_threshold_exceeded 10: Physical_layer_retransmission_timeout 11: Heartbeat_errors 12: Link_Layer_credit_monitoring_watchdog 13: Link_Layer_integrity_threshold_exceeded 14: Link_Layer_buffer_overrun 15: Down_by_outband_command_with_healthy_link 16: Down_by_outband_command_for_link_with_hi_ber 17: Down_by_inband_command_with_healthy_link 18: Down_by_inband_command_for_link_with_hi_ber 19: Down_by_verification_GW 20: Received_Remote_Fault 21: Received_TS1 22: Down_by_management_command 23: Cable_was_unplugged 24: Cable_access_issue 25: Thermal_shutdown 26: Current_issue 27: Power_budget 28: Down_by_Management_command - (e.g. BMC)	RO

Table 1192 - Link down info page Fields

Offset	Bits	Name	Description	Access
08h	7:0	remote_reason_opcode	see <i>local_reason_opcode</i>	RO
0Ch	7:0	e2e_reason_opcode	see <i>local_reason_opcode</i> for local reason opcode for remote reason opcode: <i>local_reason_opcode+100</i>	RO
10h	27:24	last_ber_mantissa		RO
	23:16	last_ber_exp		RO
	11:8	ber_mantissa		RO
	7:0	ber_exp		RO
14h	27:24	min_ber_mantissa		RO
	23:16	min_ber_exp		RO
	11:8	max_ber_mantissa		RO
	7:0	max_ber_exp		RO
18h	15:0	num_of_ber_alarms		RO

Table 1193 - Link down events page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ib_port_events																																00h
																																04h
etherent_port_events																																08h
																																0Ch
general_port_events																																10h
																																14h-1Ch

Table 1194 - Link down events Fields

Offset	Bits	Name	Description	Access
00h	31:0	ib_port_events	Bit mask for IB port causes: Bit 0: ts_at_linkup Bit 1: ibl_link_retrain Bit 2: rx_comskip_timeout Bit 3: minor_threshold_reached Bit 4: watch_dog Bit 5: excessive_buffer Bit 6: lli_err Bit 7: llr_link_retrain Bit 8: block_lock Bit 9: skip_or_COM_alignment_loss Bit 10: FEC_lock_loss Bit 11: MAD_down_command	RO
08h	31:0	ether- ent_port_events	Bit 0: remote_fault_at_linkup Bit 1: PCS_hi_ber_event Bit 2: FEC_hi_ser_event Bit 3: Block_lock_loss Bit 4: Alignment_loss Bit 5: FEC_block_lock_loss Bit 6:	RO
10h	31:0	gener- al_port_events	Bit 0: fifo_full_err Bit 1: PAOS_down_command Bit 2: disable_mask_event Bit 3: port_event - (see page 4 of PDDR) Bit 4: device_event - (see page 5 of PDDR) Bit 5: spike_detected - TBD analog capability?	RO

Table 1195 - Module Info Page Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
cable_technology								cable_breakout								ext_ethernet_compliance_code								ethernet_compliance_code								00h
cable_type				cable_vendor				cable_length								cable_identifier								cable_power_class								04h
max_power								cable_rx_amp								cable_rx_emphasis								cable_tx_equalization								08h
cable_attenuation_25g								cable_attenuation_12g								cable_attenuation_7g								cable_attenuation_5g								0Ch
cable_rx_post_emphasis								rx_cdr_cap				tx_cdr_cap				rx_cdr_state								tx_cdr_state								10h

Table 1195 - Module Info Page Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
vendor_name																																14h-20h				
vendor_pn																																24h-30h				
vendor_rev																																34h				
fw_version																																38h				
vendor_sn																																3Ch-48h				
temperature																voltage																4Ch				
rx_power_lane0																rx_power_lane1																50h				
rx_power_lane2																rx_power_lane3																54h				
rx_power_lane4																rx_power_lane5																58h				
rx_power_lane6																rx_power_lane7																5Ch				
tx_power_lane0																tx_power_lane1																60h				
tx_power_lane2																tx_power_lane3																64h				
tx_power_lane4																tx_power_lane5																68h				
tx_power_lane6																tx_power_lane7																6Ch				
tx_bias_lane0																tx_bias_lane1																70h				
tx_bias_lane2																tx_bias_lane3																74h				
tx_bias_lane4																tx_bias_lane5																78h				
tx_bias_lane6																tx_bias_lane7																7Ch				
temperature_high_th																temperature_low_th																80h				
voltage_high_th																voltage_low_th																84h				
rx_power_high_th																rx_power_low_th																88h				
tx_power_high_th																tx_power_low_th																8Ch				
tx_bias_high_th																tx_bias_low_th																90h				
																wavelength																94h				
active_set_host_compliance_code									active_set_media_compliance_code																	ib_compliance_code										98h

Table 1195 - Module Info Page Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																9Ch
																																A0h-F4h

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
00h	31:24	cable_technology	QSFP: Byte 147 per SFF-8636. SFP: SFP+ Cable Technology: byte8 per SFF-8472: Bit 3 - Active Cable Bit 2 - Passive Cable CMIS based (QSFP-DD / OSFP/ SFP-DD): Byte 212	RW
	23:16	cable_breakout	Reserved for SFP. For QSFP: Byte113 per SFF-8636 For CMIS based modules: 1 - QSFP-DD to QSFP-DD 2 - QSFP-DD to 2xQSFP or 2xQSFP-DD (depopulated / 4 lanes) 3 - QSFP-DD to 4xSFP-DD or 4xQSFP (depopulated / 2 lanes) 4 - QSFP-DD to 8xSFP 5 - QSFP-DD (depopulated / 4 lanes) or QSFP to QSFP or QSFP-DD (depopulated / 4 lanes) 6 - QSFP-DD (depopulated / 4 lanes) or QSFP to 2xQSFP(depopulated / 2 lanes) or 2xSFP-DD 7 - QSFP-DD (depopulated / 4 lanes) to 4xSFP 8 - SFP-DD to SFP-DD 9 - SFP-DD to 2xSFP Note: QSFP-DD can refer to OSFP form factor according to identifier	RW
	15:8	ext_ethernet_compliance_code	Extended Specification Compliance Codes for SFP: byte 36 per SFF-8472 for QSFP: byte192 per SFF-8636 (QSFP28) for CMIS (SFP-DD / QSFP-DD/ OSFP) : Byte 86 - Host Electrical Interface	RW

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
00h	7:0	ethernet_compliance_code	QSFP: Ethernet Compliance Codes bit mask (10/40G/100G) Byte131 per SFF-8636 Bit 7 - Extended Specification Compliance valid Bit 6 - 10GBASE-LRM Bit 5 - 10GBASE-LR Bit 4 - 10GBASE-SR Bit 3 - 40GBASE-CR4 Bit 2 - 40GBASE-SR4 Bit 1 - 40GBASE-LR4 Bit 0 - 40G Active Cable (XLPPI) SFP : 10G Ethernet Compliance Codes Byte3 per SFF-8472: Bit 7 - 10G Base-ER Bit 6 - 10G Base-LRM Bit 5 - 10G Base-LR Bit 4 - 10G Base-SR CMIS based (QSFP-DD/ SFP-DD / OSFP) Byte 87 - Module Media Interface	RW

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
04h	31:28	cable_type	Cable/module type: 0 - Unidentified 1 - Active cable (active copper / optics) 2 - Optical Module (separated). 2 - Passive copper cable 3 - Cable unplugged In CMIS based Modules per byte 85.	RO
	27:24	cable_vendor	Cable vendor: 0 - Other 1 - Mellanox 2 - Known OUI [Internal] Known OUI refers to non Mellanox OUI that has unique fw treatment (stamping, vendor paging etc.)	RO
	23:16	cable_length	Cable length in 1m units. For CMIS modules: bits 6:7 represent <i>cable_length_multiplier</i> for calculating cable length 00 - 0.1 multiplier (1 to 6.3m) 01 - 1 multiplier (1 to 63m) 10 - 10 multiplier (1 to 630m) 11 - 100 multiplier (1 to 6300m) bits 0:5 represent <i>cable_length_value</i> for calculating cable length. length is calculated with <i>cable_length_value</i> * <i>cable_length_multiplier</i>	RO
	15:8	cable_identifier	0 - QSFP28 1 - QSFP+ 2 - SFP28/SFP+ 3 - QSA (QSFP->SFP) 4 - Backplane 5 - SFP-DD 6 - QSFP-DD 7 - QSFP_CMIS 8 - OSFP 9 - C2C	RO

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
04h	7:0	cable_power_class	Module maximum power consumption for SFP/QSFP: 0 - Power Class 0 (1.0 W max) 1 - Power Class 1 (1.5 W max) 2 - Power Class 2 (2.0 W max) 3 - Power Class 3 (2.5 W max) 4 - Power Class 4 (3.5 W max) 5 - Power Class 5 (4.0 W max) 6 - Power Class 6 (4.5 W max) 7 - Power Class 7 (5.0 W max) Module maximum power consumption for SFP-DD: 0 - Power Class 0 (0.5 W max) 1 - Power Class 1 (1.0 W max) 2 - Power Class 2 (1.5 W max) 3 - Power Class 3 (2.0 W max) 4 - Power Class 4 (3.5 W max) 5 - Power Class 5 (5.0 W max) 6 - reserved 7 - reserved 8 - Power Class 8 (> 5.0 W max) Module maximum power consumption for QSFP-DD/OSFP: 1 - Power Class 1 (1.5 W max) 2 - Power Class 2 (3.5 W max) 3 - Power Class 3 (7.0 W max) 4 - Power Class 4 (8.0 W max) 5 - Power Class 5 (10 W max) 6 - Power Class 6 (12 W max) 7 - Power Class 7 (14 W max) 8 - Power Class 8 (> 14 W max)	RO
08h	31:24	max_power	Byte 201 for CMIS (QSFP-DD/ SFP-DD/ OSFP) Other Cable ignore field.	RO
	23:16	cable_rx_amp	Reserved for SFP [Internal] For multi lane form factor (i.e. QSFP) , The value represent channel 0.	RO
	15:8	cable_rx_emphasis	[Internal] For multi lane form factor (i.e. QSFP) , The value represent channel 0. for CMIS (QSFP-DD/ SFP-DD/ OSFP) field will represent Rx pre-emphasis.	RO
	7:0	cable_tx_equalization	[Internal] For multi lane form factor (i.e. QSFP) , The value represent channel 0.	RO

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
0Ch	31:24	cable_attenuation_25g	Valid only for CMIS (QSFP-DD/ SFP-DD/ OSFP) Other Cable ignore field.	RO
0Ch	23:16	cable_attenuation_12g	Reserved for SFP	RO
	15:8	cable_attenuation_7g	Reserved for SFP	RO
	7:0	cable_attenuation_5g	Reserved for SFP	RO

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
10h	31:24	cable_rx-post_emphasis	Valid only for CMIS (QSFP-DD/ SFP-DD/ OSFP) Rx post-emphasis.	RO
	23:20	rx_cdr_cap	0 - No CDR 1 - Build-in CDR with on/off control 2 - Build-in CDR without on/off control	RO
	19:16	tx_cdr_cap	0 - No CDR 1 - Build-in CDR with on/off control 2 - Build-in CDR without on/off control	RO
	15:8	rx_cdr_state	Reserved for SFP Bit 0 - RX CDR on/off on channel 0 Bit 1 - RX CDR on/off on channel 1 Bit 2 - RX CDR on/off on channel 2 Bit 3 - RX CDR on/off on channel 3 Bit 4 - RX CDR on/off on channel 4 Bit 5 - RX CDR on/off on channel 5 Bit 6 - RX CDR on/off on channel 6 Bit 7 - RX CDR on/off on channel 7 CDR on - when bit is set. CDR off - when bit is clear.	RO
	7:0	tx_cdr_state	Reserved for SFP Bit 0 - TX CDR on/off on channel 0 Bit 1 - TX CDR on/off on channel 1 Bit 2 - TX CDR on/off on channel 2 Bit 3 - TX CDR on/off on channel 3 Bit 4 - TX CDR on/off on channel 4 Bit 5 - TX CDR on/off on channel 5 Bit 6 - TX CDR on/off on channel 6 Bit 7 - TX CDR on/off on channel 7 CDR on - when bit is set. CDR off - when bit is clear.	RO
14h-20h	128	vendor_name		RO
24h-30h	128	vendor_pn	Vendor Part Number	RO
34h	31:0	vendor_rev		RO
38h	31:0	fw_version	If information is not available by the module: set to 0	RO

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
3Ch-48h	128	vendor_sn	Vendor Serial Number	RO
4Ch	31:16	temperature	module temperature in 1/256 C	RO
	15:00	voltage	Internally measured supply voltage in 100uV	RO
50h	31:16	rx_power_lane0	RX measured power channel 0 in dBm	RO
	15:0	rx_power_lane1	RX measured power channel 1 in dBm	RO
54h	31:16	rx_power_lane2	RX measured power channel 2 in dBm	RO
	15:0	rx_power_lane3	RX measured power channel 3 in dBm	RO
58h	31:16	rx_power_lane4	RX measured power channel 4 in dBm	RO
	15:0	rx_power_lane5	RX measured power channel 5 in dBm	RO
5Ch	31:16	rx_power_lane6	RX measured power channel 6 in dBm	RO
	15:0	rx_power_lane7	RX measured power channel 7 in dBm	RO
60h	31:16	tx_power_lane0	TX measured power channel 0 in dBm	RO
	15:0	tx_power_lane1	TX measured power channel 1 in dBm	RO
64h	31:16	tx_power_lane2	TX measured power channel 2 in dBm	RO
	15:0	tx_power_lane3	TX measured power channel 3 in dBm	RO
68h	31:16	tx_power_lane4	TX measured power channel 4 in dBm	RO
	15:0	tx_power_lane5	TX measured power channel 5 in dBm	RO
6C	31:16	tx_power_lane6	TX measured power channel 6 in dBm	RO
	15:0	tx_power_lane7	TX measured power channel 7 in dBm	RO
70h	31:16	tx_bias_lane0	TX measured bias current on channel 0 in 2uA	RO
	15:0	tx_bias_lane1	TX measured bias current on channel 1 in 2uA	RO
74h	31:16	tx_bias_lane2	TX measured bias current on channel 2 in 2uA	RO
	15:0	tx_bias_lane3	TX measured bias current on channel 3 in 2uA	RO
78h	31:16	tx_bias_lane4	TX measured bias current on channel 4 in 2uA	RO
	15:0	tx_bias_lane5	TX measured bias current on channel 5 in 2uA	RO
7Ch	31:16	tx_bias_lane6	TX measured bias current on channel 6 in 2uA	RO
	15:0	tx_bias_lane7	TX measured bias current on channel 7 in 2uA	RO

Table 1196 - Module Info Page Fields

Offset	Bits	Name	Description	Access
80h	31:16	tempera- ture_high_th	Alarm high temperature threshold in 1/256 C	RO
	15:0	tempera- ture_low_th	Alarm low temperature threshold in 1/256 C	RO
84h	31:16	voltage_high_th	Alarm high Voltage threshold in 100uV	RO
	15:0	voltage_low_th	Alarm low Voltage threshold in 100uV	RO
88h	31:16	rx_pow- er_high_th	Alarm high RX Power threshold in dBm. Taking only from channel 0.	RO
	15:0	rx_pow- er_low_th	Alarm low RX Power threshold in dBm. Taking only from channel 0.	RO
8Ch	31:16	tx_pow- er_high_th	Alarm high TX Power threshold in dBm. Taking only from channel 0.	RO
	15:0	tx_pow- er_low_th	Alarm low TX Power threshold in dBm. Taking only from channel 0.	RO
90h	31:16	tx_bias_high_th	Alarm high TX Bias current threshold in 2 uA.	RO
	15:0	tx_bias_low_th	Alarm low TX Bias current threshold in 2 uA.	RO
94h	15:0	wavelength	Nominal laser wavelength in nm	RO
98h	31:24	active_set_host_ compli- ance_code	Valid for CMIS modules only. According to current Active set, value of Host Electrical Interface byte	
98h	23:16	active_set_me- dia_compli- ance_code	Valid for CMIS modules only. According to current Active set, value of Module Media Interface byte	
98h	7:0	ib_compli- ance_code	Byte 164 of SFF-8636 For CMIS modules IB Protocols.	RO
9Ch	5:0	ib_width	Bitmask of width of IB Protocols	RO

Table 1197 - Troubleshooting info Page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1197 - Troubleshooting info Page layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status_message																																0Ch-F4h

Table 1198 - Troubleshooting info Page Fields

Offset	Bits	Name	Description	Access
00h	15:0	group_opcode	0 - Monitor opcodes [Internal] 1 - Advanced debug opcodes	Index
04	31:0	status_opcode	Status opcode described in: See Table 1199, “PDDR - Monitor opcodes layout,” on page 1392 [Internal] See Table 1201, “PDDR - Advanced opcodes layout,” on page 1394	RO
08h	31:16	user_feedback_data	TBD	RW
	15:0	user_feedback_index	TBD	RW
0Ch-F4h	1888	status_message	ASCII code message All Messages are terminated by a Null character ‘\0’	RO

28.19.61.9 Monitor Opcodes

Table 1199 - PDDR - Monitor opcodes layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																monitor_opcode															00h	

Table 1200 - PDDR - Monitor opcodes Fields

Offset	Bits	Name	Description	Access
00h	15:0	monitor_opcode	Status opcode: PHY FW indication (0 - 1023): 0 - No issue observed 1 - Port is close by command (see PAOS). 2,3,4,38,39 - AN failure 5,6,7,8 - Link training failure. 9,10,11,12,13 - Logical mismatch between link partners 14 - Remote fault received 15,42,17 - Bad signal integrity 16,24-32 - Cable compliance code mismatch (protocol mismatch between cable and port) [Internal] TBD - include MCAS stamping 23,22,19,18- Internal error 34,35 - Speed degradation 128 - Troubleshooting in process 1023- Info not available MNG FW issues (1024 - 2047): 1024 - Cable is unplugged 1025 - Long Range for non Mellanox cable/module . 1026 - Bus stuck (I2C Data or clock shorted) 1027 - Bad/unsupported EEPROM 1028 - Part number list 1029 - Unsupported cable. 1030 - Module temperature shutdown 1031 - Shorted cable 1032 - Power budget exceeded 1033 - Management forced down the port 1034 - Module is disabled by command 1036 - Module's PMD type is not enabled (see PMTPS). [internal]1037- Laster TEC failure [internal]1038 - High current [internal]1039- High voltage 1040 - pcie system power slot Exceeded [internal]1041- High power [internal]1042 - Module state machine fault 1043,1044,1045,1046 - Module's stamping speed degeneration Core/Driver (2048 - 3071):	RW

[Internal] Advanced debug Opcodes

Table 1201 - PDDR - Advanced opcodes layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																advanced_opcode															00h	

Table 1202 - PDDR - Advanced opcodes Fields

Offset	Bits	Name	Description	Access
00h	15:0	advanced_op-code	Status opcode: PHY FW indication (0 - 1023): 0 - No issue observed 1 - Port is close by command (see PAOS). 2 - AN no partner detected. 3 - AN ack not received. 4 - AN next-page exchange failed. 5 - KR frame lock not acquired. 6 - KR link inhibit timeout 7 - KR Link partner didn't set receiver ready 8 - KR tuning didn't completed 9 - PCS didn't acquire block lock. 10 - PCS didn't acquire AM lock (NO FEC) 11 - PCS didn't get align_status. 12 - FC FEC isn't locked 13 - RS FEC isn't locked 14 - Remote fault received 15 - Bad Signal integrity 16 - Compliance code mismatch (protocol mismatch between cable and port) 17 - Large number of physical errors (high BER). 18 - Port is disabled by Ekey 19 - Phase EO failure 20 - stamping of Non Mellanox Cables/Modules 21 - Down by PortInfo MAD 22 - Disabled by Verification 23 - Calibration failure 24 - EDR speed is not allowed due to cable stamping: EDR stamping 25 - FDR10 speed is not allowed due to cable stamping: FDR10 stamping 26 - Port is closed due to cable stamping: Ethernet_compliance_code_zero 27 - Port is closed due to cable stamping: 56GE stamping 28 - Port is closed due to cable stamping: non-Mellanox QSFP28 29 - Port is closed due to cable stamping: non-Mellanox SFP28 30 - Port is closed, no backplane enabled speed over backplane channel 31 - Port is closed, no passive protocol enabled over passive copper channel 32 - Port is closed, no active protocol enabled over active channel 33 - Port width is doesn't match the port speed enabled 34 - Local speed degradation 35 - Remote speed degradation 36 - No Partner detected during force mode. 37 - Partial link indication during force mode. 38 - AN Failure - FEC mismatch during override 39 - AN Failure - No HCD 40 - VPI protocol don't match 41 - Port is closed, module can't be set to the enabled rate 42 - Bad SI, cable is configured to non optimal rate 43 - No Partner detected during force mode, while port is running fast link-up 44 - Partial link indication during force mode, while port is running fast link-up 45 - Main USR logic didn't get align_status. 46 - Tile USR logic didn't get align_status; 128 - Troubleshooting in process 1023 - Info not available MNG FW issues (1024 - 2047): 1024 - Cable is unplugged/powered off 1025 - Long Range for non MLNX cable/module . 1026 - Bus stuck (I2C Data or clock shorted) 1027 - bad/unsupported EEPROM 1028 - part number list 1029 - unsupported cable. 1030 - module temperature shutdown 1031 - Shorted cable 1032 - Power Budget Exceeded 1033 - Management force down the port 1034 - Module is disabled by command 1035 - System Power is Exceeded therefore the module is powered off. 1036 - Module's PMD type is not enabled (see PMTPS). 1043 - NDR speed is not allowed due to cable stamping: NDR stamping 1044 - HDR speed is not allowed due to cable stamping: HDR stamping 1045 - EDR speed is not allowed due to cable stamping: EDR stamping 1046 - FDR10 speed is not allowed due to cable stamping: FDR10 stamping Core/Driver (2048 - 3072):	RW

Table 1204 - Port events Page Fields

Offset	Bits	Name	Description	Access
00h	31	invalid_fsm_sv	invalid_fsm_ps severity 0 - misconfiguration 1 - fatal	RO
	15:0	invalid_fsm_ps	Bit mask per FSM:	RO
04h	17	kr_false_ready_sv	KR startup false remote ready severity	RO
	16	kr_false_ready	KR startup false remote ready indication	RO
	11	mod_req_nack_sv	TBD - move to per port	RO
	10	mod_req_nack		RO
	9	mod_req_busy_sv		RO
	8	mod_req_busy		RO
	7	mod_config_to_sv	Module configuration timeout severity	RO
	6	mod_config_to	Module configuration timeout indication	RO
	5	mod_adapt_faild_sv	Module adaptive tuning failed severity	RO
	4	mod_adapt_faild	Module adaptive tuning failed indication	RO
	3	pm_fifo_full_sv	Phy Manager fifo full severity	RO
	2	pm_fifo_full	Phy Manager fifo full indication	RO
	1	protocol_check_sv	illegal_protocol severity	RO
	0	protocol_check	illegal_protocol indication	RO
08h	31	input_event_sv	invalid_fsm_ps severity 0 - misconfiguration 1 - fatal	RO
	15:0	input_event	Bit mask per input event	RO

Table 1204 - Port events Page Fields

Offset	Bits	Name	Description	Access
0Ch	3	lane_map- ping_sv		RO
	2	lane_mapping		RO
	1	qsfp_zero_at- ten_sv	QSFP zero attenuation severity 0 - misconfiguration 1 - fatal	RO
	0	qsfp_zero_atten	QSFP zero attenuation indication	RO
10h	9	mono_reach_lo w_limit_sv	mono_reach_low_limit severity 0 - misconfiguration 1 - fatal	RO
	8	mono_reach_lo w_limit		RO
	7	mono_reach_hig h_limit_sv	mono_reach_high_limit severity 0 - misconfiguration 1 - fatal	RO
	6	mono_reach_hig h_limit		RO
	5	mono_- flow_height_sv	mono_flow_height severity 0 - misconfiguration 1 - fatal	RO
	4	mono_- flow_height		RO
	3	sin- gle_gains_bad_v al_sv	single_gains_bad_val severity 0 - misconfiguration 1 - fatal	RO
	2	sin- gle_gains_bad_v al		RO
	1	init_all_gains_ba d_val_sv	init_all_gains_bad_val severity 0 - misconfiguration 1 - fatal	RO
	0	init_all_gains_ba d_val	init_all_gains_bad_val	RO

Table 1204 - Port events Page Fields

Offset	Bits	Name	Description	Access
20h-3Ch	256	clr_mask	Valid only when PDDR.clr is set Bit mask for page bits 255:0, when bits 31:0 assigned to offset 00h bits 31:0. bits 63:32 assigned to offset 04h bits 31:0. bits 95:64 assigned to offset 08h bits 31:0. bits 127:69 assigned to offset 0Ch bits 31:0. Only when PDDR.clr is set and the appropriate bit in the mask is set, the event will be cleared. For example: To clear the kr_false_ready need set PDDR.clr , select the “port event page” and set bit 48 in clr_mask.	OP

Table 1205 - Device events Page Layout

Offset	00h	04h	08h	0Ch
0	invalid_port_access	pll_state_mask_center0	varactors_calib_fail_center0	irisc_stuck_mask
1			varactors_calib_fail_center0_sv	
2			varactors_calib_fail_center1	
3			varactors_calib_fail_center1_sv	
4		logical_phy_to		
5		logical_phy_to_sv		
6		analog_phy_to		
7		analog_phy_to_sv		
8		pll_state_sv_0		
9				
10				
11				
12				
13				
14				
15				
16	pll_state_mask_center1	cr_space_timeout		
17		cr_space_timeout_sv		
18				
19				
20				
21				
22				
23				
24		pll_state_sv_1		
25				
26				
27				
28				
29				
30				
31	invalid_port_access_sv			irisc_stuck_mask_sv

Table 1206 - device events Page Fields

Offset	Bits	Name	Description	Access
08h	17	cr_space_time-out_sv		RO
	16	cr_space_time-out		RO
	7	analog_phy_to_sv	SerDes timeout severity	RO
	6	analog_phy_to	SerDes timeout indication	RO
	5	logical_phy_to_sv	PLU timeout severity	RO
	4	logical_phy_to	PLU timeout indication	RO
	3	varactors_calib_fail_center1_sv		RO
	2	varactors_calib_fail_center1		RO
	1	varactors_calib_fail_center0_sv		RO
	0	varactors_calib_fail_center0		RO
0Ch	31	irisc_stuck_mask_sv		RO
	23:0	irisc_stuck_mask		RO
14h	24	pll_lock_sv_1		RO
	19:16	pll_unlock_mask_center1		RO
	8	pll_lock_sv_0		RO
	3:0	pll_unlock_mask_center0		RO

Table 1206 - device events Page Fields

Offset	Bits	Name	Description	Access
18h	17	pwr_governor_err_sv		RO
	16	pwr_governor_err		RO
	5	timer_error_sv	TBD	RO
	4	timer_error	TBD	RO
	3	iopl_err_center1_sv		RO
	2	iopl_err_center1		RO
	1	iopl_err_center0_sv		RO
	0	iopl_err_center0		RO
20h-3Ch	256	clr_mask	Valid only when PDDR.clr is set Bit mask for page bits 255:0, when bits 31:0 assigned to offset 00h bits 31:0. bits 63:32 assigned to offset 04h bits 31:0. bits 95:64 assigned to offset 08h bits 31:0. bits 127:69 assigned to offset 0Ch bits 31:0. Only when PDDR.clr is set and the appropriate bit in the mask is set, the event will be cleared	OP

Table 1207 - Link Up Info Page Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
												up_rea- son_pwr								up_rea- son_drv								up_rea- son_mng				00h
																time_to_link_up																04h
																													fast_link_up _status		08h	
																																0Ch-F4h

Table 1208 - Link Up info Page Fields

Offset	Bits	Name	Description	Access
00h	19:16	up_reason_pwr	invalid port access severity 0 - No down command / unknown 1 - Keep link up on boot 2 - Keep link up Eth/IB 3 - Keep link up on standby	RO
	11:8	up_reason_drv	invalid port access severity 0 - No down command / unknown 1 - Up by at least one of the hosts	RO
	3:0	up_reason_mng	invalid port access severity 0 - No down command / unknown 1 - BMC force linkup 2 - WOL force linkup 3 - ASN.1 force link up	RO
04h	15:0	time_to_link_up	Time in msec.	RO
08h	3:0	fast_link_up_status	Indicates if fast link-up was performed in the link: 0: Unknown 1: Fast Link-UP was performed. 2: Regular link-up flow performed due to changes in cable 3: Regular link-up flow performed due to changes in speed 4: Regular link-up performed due to changes in temperature 5: Regular link-up performed due to MLPN flow 6: Regular link-up performed due to PCNR.tuning_override configuration while port was in “down” state. 7: Regular link-up performed due to changes in TX rev of remote side 8: Fast Link-up was performed, but the total link-up time is expected to be long due to changes in TX rev of local side (i.e. remote doesn't activate fast link-up flow)	RO

28.19.62PPBMP - Port Phy BER Monitor Parameters

The register enables to read and set the BER Monitor parameters. The BER Monitor parameters are set per device. [Internal] while the monitoring can be enabled on a per port basis using XXXX register. When the BER monitor parameters during ports operation, they are immediately applied.

Table 1209 - PPBMP - Port Phy BER Monitor Parameters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
								local_port								pnat									port_type				monitor_group			00h				
monitor_parameters																																04h-08h				
																																0Ch-2Ch				

Table 1210 - PPBMP - Port Phy BER Monitor Parameters Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type. determines the way <i>local_port</i> is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index
	3:0	monitor_group	Monitor Parameters Group 0x0: Pre-FEC BER monitor parameters for RS-FEC links 0x1: Pre-FEC BER monitor parameters for FC-FEC links 0x2: Effective BER (post FEC/NO-FEC) monitor parameters [internal] 0x4: Pre-FEC BER monitor parameters for ZL-FEC link Note: in USR port only ZL-FEC and post FEC/NO-FEC are possible	Index
04h-08h	64	monitor_parameters	See Table 1211, “PPBMP - Monitor Parameters Layout,” on page 1405	RW

Table 1211 - PPBMP - Monitor Parameters Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				alarm_th_mantissa				alarm_th_exp												warning_th_mantissa				warning_th_exp								00h
				normal_th_mantissa				normal_th_exp																								04h

Table 1212 - PPBMP - Monitor Parameters Fields

Offset	Bits	Name	Description	Access
00h	27:24	alarm_th_mantissa	BER threshold for alarm state, when the monitored BER crosses above the alarm_threshold , the monitor state is set to alarm. alarm_threshold = alarm_th_mantissa * 10 ^{^(-alarm_th_exp)} The alarm_threshold shall be within the supported range for the monitor type	RW
	23:16	alarm_th_exp		RW
	11:8	warning_th_mantissa	BER threshold for warning state, when the monitor state is normal and the monitored BER crosses above the warning_threshold , the monitor state will be set to warning warning_threshold = warning_th_mantissa * 10 ^{^(-warning_th_exp)} The warning_threshold shall be within the supported range for the monitor type	RW
	7:0	warning_th_exp		RW
04h	27:24	normal_th_mantissa	BER threshold for normal state, when the monitored BER crosses below the normal_threshold , the monitor state is set to normal. normal_threshold = normal_th_mantissa * 10 ^{^(-normal_th_exp)} The warning_threshold shall be within the supported range for the monitor type	RW
	23:16	normal_th_exp		RW

28.19.62.1 Supported Monitor Parameters per monitor type

The configured Monitor Parameters shall be within the ranges defined below. The Monitored Parameters are initially set to default values that reflect desired links behavior, it is recommended to use the default value.

Supported Monitor Parameters per monitor type

Parameter	Unit	Monitor Type	Min Value	Max Value
alarm_threshold, warning_threshold, normal_threshold¹	errors / bits	RS-FEC Pre-FEC BER Monitor	1E-9	1E-4
		FC-FEC Pre-FEC BER Monitor	1E-11	1E-4
		Effective-BER Monitor	1E-14	1E-8

Parameter	Unit	Monitor Type	Min Value	Max Value
		[internal] ZL-FEC pre-FEC channel monitor	1E-12	1E-10

1. **_th_mantissa* can be set to values between 1-9.

For all Monitor Types the following conditions must be meet:

1. *alarm_threshold* > *warning_threshold* > *normal_threshold*.

28.19.63PPBMC - Port Phy BER Monitor Control

The register enables to read and set the BER Monitor parameters. The BER Monitor parameters are set per device. [internal] while the monitoring can be enabled on a per port basis using XXXX register. When the BER monitor parameters during ports operation, they are immediately applied.

Table 1213 - PPBMC - Port Phy BER Monitor Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port								pnat				monitor_type				port_type								00h
e		ievent		event_ctrl								monitor_cntl												monitor_state								04h
																								peer_monitor_state								08h
																																0Ch

Table 1214 - PPBMC - Port Phy BER Monitor Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	monitor_type	Monitor type based on monitor_cntl setting and the operational mode of the link: 0 - Post FEC / No FEC BER monitoring 1 - Pre FEC BER monitoring	RO
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved. Configuration 4,5 should use the same clock	Index

Table 1214 - PPBMC - Port Phy BER Monitor Control Register Fields

Offset	Bits	Name	Description	Access
04h	31:30	e	Event generation on operational state change: 0 - Do not generate event 1 - Generate Event 2 - Generate Single Event Not Supported for HCA.	RW
	29	ievent	Invoke event 0: Read only 1: Does both reading and invoke an event of type PPBME_TRAP Note: setting this field invokes an event, regardless to <i>e</i> and <i>event_ctl</i> fields, it has no effect on the event state.	OP
	27:24	event_ctl	Bitmask - event generation on operational state change: Bit 0 - Normal state Bit 1 - Warning state Bit 2 - Alarm state Bit 3 - Reserved Note: Normal state events can be generated only if either Alarm and/or Warning events are enabled.	RW
	19:16	monitor_cntl	Bitmask - monitor options: Bit 0 - Pre FEC monitor: When set, a link with FEC active will monitor Pre-FEC BER. When cleared, link with FEC active will monitor Post-FEC (effective) BER. When FEC is not active, will monitor only effective BER.	RW
	7:0	monitor_state	The monitor state of the port: 0 - No BER Monitor support 1 - Normal 2 - Warning 3 - Alarm * The init monitor state for each port upon link up is normal	RO
08h	7:0	peer_monitor_state	[internal] The peer monitor state of the port: 0 - No peer BER Monitor indication support 1 - Normal 2 - Warning 3 - Alarm	RO

28.19.6 PPBME - Port Phy BER Monitor Event

The PPBME register monitor state change of a port. A change in a ports monitor state will generate an event only if the monitor has transitioned to a state that was armed prior for the transition. Software can query the PPBME register for the port monitor status

Table 1215 - PPBME - Port Phy BER Monitor Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port								pnat				monitor_type		port_type										00h
		ievent																														04h
																																08h
																																0Ch

Table 1216 - PPBME - Port Phy BER Monitor Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	11:8	monitor_type	Monitor type based on monitor_options setting and the operational mode of the link: 0 - Post FEC / No-FEC BER monitoring 1 - Pre FEC BER monitoring	RO
	7:4	port_type	For HCA supported only when indicated by PCAM 0 - Network Port 1 - Near-End Port (For Retimer/Gearbox - Host side) 2 - Internal IC LR Port 3 - Far-End Port (For Retimer/Gearbox - Line side) 4 - USR Main 5 - USR Tile Other values are reserved.	Index
04h	29	ievent	Invoke event 0: Read only 1: Does both reading and invoke an event of type PPBME_TRAP Note: setting this field invokes an event, regardless to e and event_ctl fields, it has no effect on the event state.	OP
	7:0	monitor_state	The monitor state of the port: 0 - No BER Monitor on the port 1 - Normal 2 - Warning 3 - Alarm	RO

Table 1216 - PPBME - Port Phy BER Monitor Control Register Fields

Offset	Bits	Name	Description	Access
08h	7:0	peer_monitor_state	internal] The peer monitor state of the port: 0 - No peer BER Monitor indication support 1 - Normal 2 - Warning 3 - Alarm	RO

28.19.65[Internal] PHFR - Port Halt Fsm Register

The PHFSM register is a verification register used to halt/failed a specific transition to configurable state of Port's FSMs.

Table 1217 - PHFR - Port Halt Fsm Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
								local_port								pnat										status								00h
								fsm_num																		sub_fsm_num								04h
								halt_state																		prev_state								08h
																																0Ch		
																																10h		
																																14h		
																																18h-1Ch		

Table 1218 - PHFR - Port Halt Fsm Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB / label port number [Internal] 3 - Out of band / PCI	Index
	7:0	status	0 - No halt / Failure 1 - FSM in halt as configured 2 - FSM in fault as configured	RO
04h	23:16	fsm_num	FSM number A value of 0xFF - halt all FSMs	Index
	7:0	sub_fsm_num	Sub FSM number	Index
08h	23:16	halt_state	State that will be halt / failure according to <i>halt_mode</i> value	RW
	7:0	prev_state	Previous state before the <i>halt_state</i> , according to <i>halt_mode</i> value Reserved for <i>halt_mode</i> = 0 / 1	RW
10h	4	halt_fault	Halt or Failure mode: 0 - Halt 1 - Fault	RW
	3:0	halt_mode	halt action configuration: 0 - No Halt / Failure operation 1 - Halt / Failure upon any transition to " <i>halt_state</i> " from any state. 2 - Halt / Failure upon " <i>prev_state</i> " transition to " <i>halt_state</i> ", halt the " <i>halt_state</i> " 3 - Halt / Failure upon " <i>prev_state</i> " logic that should cause a transition to " <i>halt_state</i> ", halt the " <i>prev_state</i> "	RW
14h	7:0	fault_state	The state which will be used in case of " <i>halt_fault</i> " = 1 and the halt condition occurs.	RW

28.19.66[Internal] PNLB - Ports Negotiation Logger Binding Register

The PNLB activate/de-activate Negotiation Logger engines per port.

Can be set only when *PAOS.admin_status* is down.

Table 1219 - PNLB - Ports Negotiation Logger Binding Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
logger_active								local_port								pnat				lane								status				00h
																																04h
																																08h-0C'h

Table 1220 - PNLB - Ports Negotiation Logger Binding Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	logger_active	Opcode: 0 - De-activate logger on <i>local_port</i> (default) 1 - Activate logger on <i>local_port</i>	RW
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type: 0 - Local port number 1 - IB port number	Index
	11:8	lane	physical lane number. Applicable only to KR logger.	Index
	3:0	status	Status: 0 - Good 1 - No logger resources	RO
04h	9:8	kr_rcv_xmit	Log of received and/or xmit pages Bit 0 - Rcv Bit 1 - Xmit	RW
	1:0	an_rcv_xmit	Log of received and/or xmit pages Bit 0 - Rcv Bit 1 - Xmit	RW

28.19.67[Internal] PNLR - Ports Negotiation Logger Reading Register

The PNLR allows reading a the negotiation logger agent.

Table 1221 - PNLR - Ports Negotiation Logger Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sampled								local_port								pnat				lane						an_rx_tx	kr_rx_tx	kr_type			00h	
																																04h
																																08h
								kr_page_num															an_page_num								0Ch	
an_logger (See nTable 1223, “AN logger Data Layout,” on page 1414)																																10h-5Ch
																																60h-6Ch
kr_logger (See Table 1225, “KR logger Data Layout,” on page 1415)																																70h-ACh

Table 1222 - PNLR - Ports Negotiation Logger Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	sampled	Logger valid status: 0 - No valid information 1 - Valid information	RO
	23:16	local_port	local_port	Index
	15:14	pnat	Port number access type: 0 - Local port number 1 - IB port number	Index
	11:8	lane	physical lane number. Applicable only to KR logger.	Index
	5	an_rx_tx	Log of received and/or xmit pages 0 - Rcv 1 - Xmit	Index
	4	kr_rx_tx	Log of received and/or xmit pages 0 - Rcv 1 - Xmit	Index
	3:0	kr_type	KR startup Clause: 0 - CL 72/92/93 1 - CL 136/137	RO
0Ch	23:16	kr_page_num	number of kr-training pages	RO
	7:0	an_page_num	number of AN pages	RO
10h-5Ch	640	an_logger (See Table 1223, "AN logger Data Layout," on page 1414)	See Table 1223, "AN logger Data Layout," on page 1414	RO
70h-ACh	512	kr_logger (See Table 1225, "KR logger Data Layout," on page 1415)	See Table 1225, "KR logger Data Layout," on page 1415	RO

28.19.67.1 AN Logger

Table 1223 - AN logger Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
base_page_high																															00h	
base_page_low																															04h	

Table 1223 - AN logger Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
consortium_page_high																																08h
consortium_page_low																																0Ch
next_page[0]																																10h-14h
next_page[1]																																18h-1Ch
...																																...
next_page[7]																																48h-4Ch

Table 1224 - AN Logger Data Fields

Offset	Bits	Name	Description	Access
00h	31:0	base_page_high		RO
04h	31:0	base_page_low		RO
08h	31:0	consortium_page_high		RO
0Ch	31:0	consortium_page_low		RO
10h-4Ch	64	next_page[8]		RO

28.19.67.2 KR Logger

Table 1225 - KR logger Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
page[0]																																00h
page[1]																																04h
...																																...
page[15]																																3Ch

Table 1226 - KR logger Data Fields

Offset	Bits	Name	Description	Access
00h-3Ch	31:0	page[16]		RO

28.19.68PGUID- Port GUID Register

The PGUID register returns system, node and physical port's GUID.

Table 1227 - PGUID - Port GUID Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
local_port																																04h-10h
																sys_guid																14h-20h
																node_guid																24h-30h
																port_guid																34h-40h
																allocated_guid																44h-5Ch

Table 1228 - PGUID- Port GUID Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	local_port number	Index
	15:14	pnat	Port number access type. determines the way local_port is interpreted: 0 - Local port number 1 - IB port number	Index
04h-10h	128	sys_guid	System GUID	RO
14h-20h	128	node_guid	Node GUID	RO

Table 1228 - PGUID- Port GUID Register Fields

Offset	Bits	Name	Description	Access
24h-30h	128	port_guid	Port GUID	RO
34h-40h	128	allocated_guid	Allocated GUID	RO

28.19.69PMTM - Port Module Type Mapping Register

The PMTM allows query or configuration of module types.

PMTM can be set only when module is disabled by PMAOS.

Table 1229 - PMTM - Port Module Type Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch

Table 1230 - PMTM - Port Module Type Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	module number	Index
04h	12:8	module_width	Number of module's electrical lanes NOTE: For SET command, this value should match the module type width in <i>module_type</i> field	RW
	4:0	module_type	module_type: 0: Backplane_with_4_lanes 1: QSFP 2: SFP 4: Backplane_with_single_lane 8: Backplane_with_two_lanes 10: Chip2Chip4x 11: Chip2Chip2x 12: Chip2Chip1x 14: QSFP_DD 15: OSFP 16: SFP_DD 17: DSFP 18: Chip2Chip8x	RW

28.19.70[Internal] PLAR - Port Local Allocation Register

The PLAR provides information about which local port are allocated.

Notice: When auto-split feature is implemented races conditions may occur when querying PLAR without the appropriate stabilization time after inserting the cable.

Table 1231 - PLAR- Port Local Allocation Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
local_port_allocated																																00h-3Ch
																																40h-7Ch

Table 1232 - PLAR- Port Local Allocation Register Fields

Offset	Bits	Name	Description	Access
00h-3Ch	512	local_port_allocated	A bit vector where each bit represents a local port (bit 0 represents local port 1 etc). A set bit indicates that this local port is been allocated and mapped (by PSPA and PMLP).	RO

28.20 Congestion Control Registers

The following section lists the various registers used to configure/query the QoS parameters of the **adapter**.

Table 1233 summarizes Switch register set for Congestion Control.

Table 1233 - Congestion Control Register Summary

Register Name	Register ID (hex)	Query	Write	InfiniBand	Ethernet	HCA	Reference
CWGCR	0x2801	v	v	-	-		Table 1234, “CWGCR - Congestion WRED ECN General Configuration Register Layout,” on page 1423
CWTP	0x2802	v	v	-	v	v	Table 1236, “CWTP - Congestion WRED ECN TClass Profile Register Layout,” on page 1425
CWPP	0x2804	v	v	-	-		Table 1238, “CWPP - Congestion WRED ECN Pool Profile Register Layout,” on page 1427
CWTPM	0x2803	v	v	-	v	v	Table 1240, “CWTPM - Congestion WRED ECN TClass and Pool Register Layout,” on page 1428
CPQE	0x2806	v	v	-	-		Table 1242, “CPQE - Congestion Phantom Queue ECN Register Layout,” on page 1429
CHLTR	0x2810	v	v	-	-		Table 1244, “CHLTR - Congestion High Latency Table Register Layout,” on page 1430
CHLTM	0x2811	v	v	-	-		Table 1246, “CHLTM - Congestion High Latency Tclass Mapping Register Layout,” on page 1431
CHLMM	0x2812	v	v	-	-		Table 1248, “CHLMM - Congestion High Latency Multicast Mapping Register Layout,” on page 1432
CNCT	0x2030	v	v	-	-		Table 1250, “CNCT - Congestion Notification Control Register Layout,” on page 1433
CPCS	0x2031	v	v	-	-		Table 1252, “CPCS - Congestion Point Control and Status Register Layout,” on page 1433
CPID	0x2032	v	-	-	-		Table 1254, “CPID - Congestion Point Identifier Register Layout,” on page 1435
CNMC	0x2033	v	v	-	-		Table 1256, “CNMC - Congestion Notification Message Configuration Register Layout,” on page 1436
CEGCR	0x2820	v	v	-	-		Section 28.20.13, “CEGCR - Congestion Elephants Global Configuration Register,” on page 1437
CEPC	0x2821	v	v	-	-		Section 28.20.14, “CEPC - Congestion Elephants Port Configuration Register,” on page 1439

Table 1233 - Congestion Control Register Summary

Register Name	Register ID (hex)	Query	Write	InfiniBand	Ethernet	HCA	Reference
CEDR	0x2822	v	v	-	-		Section 28.20.15, “CEDR - Congestion Elephants Detection Register,” on page 1440
CEER	0x2823	v	-	-	-		Section 28.20.16, “CEER - Congestion Elephants Entry Register,” on page 1441
CWCAM	0x281F	-	-	-	v	v	Section 1266, “CWCAM - Congestion WRED Capabilities Mask Register Layout,” on page 1447

28.20.1 CWGCR - Congestion WRED ECN General Configuration Register

The CWGCR register sets global configuration.

Table 1234 - CWGCR - Congestion WRED ECN General Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
															scd																	00h
																																04h
																																08h
																																0Ch
																																10h

Table 1235 - CWGCR - Congestion WRED ECN General Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	27:24	en	enable bits: bit2: WRED drop enable For Spectrum, ingress mirror of WRED can utilize this configuration, see Section 4.12.2, “MPAR - Monitoring Port Analyzer Register,” on page 698	RW
	16	scd	Enable ECN Source Congestion Detection only. Don't mark victim packets. [Internal] source congestion is detected according to pause	RW
	11:8	aqs_weight	Average queue size weight. The average queue size (aqs) is calculated by: $aqs = ((current\_queue\_size) * 2^{-w}) + ((previous\_aqs) * (1 - 2^{-w}))$ where w is aqs_weight The time between current_queue_size and previous_aqs is configured by aqs_time Note: when mode =percentage then there is no moving average (the aqs_weight is effectively 0). See Section 28.20.2, “CWTP - Congestion WRED ECN TClass Profile Register,” on page 1425	RW
	7:0	aqs_time	Average queue size time Time from current_queue_size to previous_aqs The time is $2^{aqs\_time} * 64nSec$ Valid range is 0..24 [Internal] HW issue: for values 0..K the low pass filter may fake a little - response may be slower in case of an empty queue. Spectrum: K=10 Spectrum-2: K=11 (*2 ports) This can be fixed by a configuration but then shared-buffer egress response will be slower (doubled)	RW

Table 1235 - CWGCR - Congestion WRED ECN General Configuration Register Fields

Offset	Bits	Name	Description	Access
04h	1	mece	Mirror ECN for Congestion Experienced 0: do not ecn-mirror a packet which has Congestion Experienced (ECN=11 at ingress) 1: do ecn-mirror for a packet which has Congestion Experienced (ECN=11 at ingress) and experiences congestion in the switch (default) In any case, a packet coming with ECN=00 is not ecn-mirrored Reserved when Spectrum-1 Reserved when Quantum which always does ecn-mirror regardless of fecn in the packet	RW
04h	0	cece	Count ECN for Congestion Experienced 0: do not count "ECN mark" for a packet which has Congestion Experienced (ECN=11 at ingress) 1: do count ECN mark for a packet which has Congestion Experienced (ECN=11 at ingress) and experiences congestion in the switch (default) In any case, a packet coming with ECN=00 is not counted as ECN.	RW

28.20.2 CWTP - Congestion WRED ECN TClass Profile Register

This CWTP register configures the profiles for queues of egress port and traffic class.

Table 1236 - CWTP - Congestion WRED ECN TClass Profile Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port								pnat																00h
																																04h-1Ch
																																20h
																																24h
																																28h
																																2Ch
																																30h
																																34h
																																38h
																																3Ch

Table 1237 - CWTP - Congestion WRED ECN TClass Profile Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number CPU port is not supported	Index
	15:14	pnat	Port number access type. Determines the way local_port is interpreted: 2: host_port_number	Index
20h	7:0	traffic_class	Traffic Class Range is 0.. cap_max_tclass_sb -1	Index
24h	24	mode	Mode of thresholds for WRED and ECN: 0: Fixed 1: Percentage	RW

Table 1237 - CWTP - Congestion WRED ECN TClass Profile Register Fields

Offset	Bits	Name	Description	Access
28h	19:0	profile1_min	Minimum Average Queue Size of the profile number <i> When mode is Fixed: Cells. Valid inputs are 1.. <i>cap_total_buffer_size/cap_cell_size</i> . For Spectrum: must be a multiple of 64 Note: for Spectrum, must be larger than 2^ <i>aqs_weight</i> cells, see Section 28.20.1, “CWGCR - Congestion WRED ECN General Configuration Register,” on page 1423 When mode is Percentage: Percentage of the quota of <i>local_port,traffic_class</i> . Range is 0..100, units of integer percentage. When ECN Source Congestion Detection is enabled then congestion occurs when there are at least 256 cells in the buffer [Internal] for shared buffer dynamic, percentage is from free*alpha	RW
2Ch	30:24	profile1_percent	Percentage of WRED and ECN marking for Maximum Average Queue Size of the profile number <i> Range is 0..100, units of integer percentage Default is 0	RW
	19:0	profile1_max	Maximum Average Queue Size of the profile number <i> When mode is Fixed: Cells. Valid inputs are 1.. <i>cap_total_buffer_size/cap_cell_size</i> For Spectrum: must be a multiple of 64 When mode is Percentage: Percentage of the quota of <i>local_port,traffic_class</i> . Range is 0..100, units of integer percentage. profile<i>_max must always be equal or greater than profile<i>_min	RW
30h	19:0	profile2_min	Same as above.	RW
34h	30:24	profile2_percent	Same as above.	RW
	19:0	profile2_max	Same as above.	RW
38h	19:0	profile3_min	Same as above.	RW
3Ch	30:24	profile3_percent	Same as above.	RW
	19:0	profile3_max	Same as above.	RW

28.20.3 CWPP - Congestion WRED ECN Pool Profile Register

The CWPP register configures the profiles for all queues in shared pool.

Table 1238 - CWPP - Congestion WRED ECN Pool Profile Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h
																																14h
																																18h
																																1Ch

Table 1239 - CWPP - Congestion WRED ECN Pool Profile Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	pool	Egress pool in the switch shared buffer Range is 0.. <i>cap_num_pool_supported</i> -1	Index
08h	19:0	profile1_min	Minimum Average Queue Size of the profile number <i>, in cells. Valid inputs are 0.. <i>cap_total_buffer_size/cap_cell_size</i> For Spectrum: must be a multiple of 64	RW
0Ch	30:24	profile1_per- cent	Percentage of WRED and ECN marking for Maximum Average Queue Size of the profile number <i> Range is 0..100, units of integer percentage Default is 0	RW
	19:0	profile1_max	Maximum Average Queue Size of the profile number <i>, in cells. Valid inputs are 0.. <i>cap_total_buffer_size/cap_cell_size</i> For Spectrum: must be a multiple of 64	RW
10h	19:0	profile2_min	Same as above.	RW
14h	30:24	profile2_per- cent	Same as above.	RW
	19:0	profile2_max	Same as above.	RW
18h	19:0	profile3_min	Same as above.	RW
1Ch	30:24	profile3_per- cent	Same as above.	RW
	19:0	profile3_max	Same as above.	RW

28.20.4 CWTPM - Congestion WRED ECN TClass and Pool Mapping Register

The CWTPM register maps each egress port and traffic class to profile number. Both the Tclass profile number and the pool profile number are assigned.

Table 1240 - CWTPM - Congestion WRED ECN TClass and Pool Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								local_port								pnat																	00h
																																04h-1Ch	
																								traffic_class									20h
																														ew	ec	24h	
																																28h-30h	
																																34h	
																																38h	
																																3Ch	
																																40h	

Table 1241 - CWTPM - Congestion WRED ECN TClass and Pool Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number CPU port is not supported	Index
	15:14	pnat	Port number access type. Determines the way local_port is interpreted: 2: host_port_number	Index
20h	7:0	traffic_class	Traffic Class Range is 0.. <i>cap_max_tclass_sb</i> -1	Index
24h	1	ew	Enable WRED on traffic class: 0: Disable 1: Enable Default is 0 Reserved when Quantum	RW
	0	ee	Enable ECN on traffic class 0: Disable 1: Enable Default is 0 Reserved when Quantum	RW

28.20.5 CPQE - Congestion Phantom Queue ECN Register

The CPQE register configures the ECN marking rate for phantom queues. Note that ECN marking due to buffer state runs in parallel

*** *The Phantom Queue register is not supported by current FW versions* ***

Table 1242 - CPQE - Congestion Phantom Queue ECN Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1243 - CPQE - Congestion Phantom Queue ECN Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Egress Local Port	Index
08h	22:16	marking_percent	ECN Marking percentage when the <i>local_port</i> rate is above the phantom queue rate as configured by <i>phantom_queue_rate</i> in QEEC, see Section 19.3.10, “QEEC - QoS ETS Element Configuration,” on page 954 Range is 0..100, units of integer percentage	RW

28.20.6 CHLTR - Congestion High Latency Table Register

The CHLTR register configures the Latency Table.

For Spectrum-2 the latency threshold is used for latency-mirroring.

[Internal] For Spectrum-3 may also support `latency_traps` and `buffer_snapshot` trigger

Table 1244 - CHLTR - Congestion High Latency Table Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1245 - CHLTR - Congestion High Latency Table Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	hlt_table_index	High Latency Table index Range 0 .. <i>cap_max_high_latency_table</i> - 1 [Internal] for Quantum: <i>cap_max_high_latency_table</i> = 1	Index
08h	19:0	high_latency_thr	High latency threshold Units of 1uSec Note: HW has latency measurement error of $\pm 250\text{nSec}$ Note: packets with a latency larger than 4Sec latency may not be latency-mirrored. Thus it is recommended to set SLL to 4Sec [Internal] HW has a resolution , FW hard coded as 1uSec. Packets with a latency larger than ($4 \times 10^6 \times \text{resolution}$) latency may not be latency-mirrored. [Internal] Measures latency from DQA to DCM while the latency field in the mirror header is end-to-end. That's why it is written that "measurement error of $\pm 250\text{nSec}$ "	RW

28.20.7 CHLTM - Congestion High Latency Tclass Mapping Register

The CHLTM register configures the mapping of Tx_port.Tclass to High Latency Table.

For Spectrum-2 this is only for Unicast packets

For Quantum this is both for Unicast and for Multicast Packets.

Table 1246 - CHLTM - Congestion High Latency Tclass Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h

Table 1247 - CHLTM - Congestion Latency Tclass Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number CPU port is not supported	Index
04h	7:0	traffic_class	Traffic Class Range is 0.. <i>cap_max_tclass_data</i> -1	Index
10h	7:0	hlt_table_pointer	High Latency Table pointer for unicast packets Range 0 .. <i>cap_max_high_latency_table</i> - 1 <i>[Internal] for Quantum: cap_max_high_latency_table = 1</i>	RW

28.20.8 CHLMM - Congestion High Latency Multicast Mapping Register

The CHLMM register configures the mapping of switch_prio to High Latency Table for multi-cast packets.

Table 1248 - CHLMM - Congestion High Latency Multicast Mapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h-0Ch
																																10h

Table 1249 - CHLMM - Congestion Latency Multicast Mapping Register Fields

Offset	Bits	Name	Description	Access
00h	11:8	switch_prio	Switch Priority	Index
10h	7:0	hlt_table_pointer	High Latency Table pointer for multicast packets Range 0 .. <i>cap_max_high_latency_table</i> - 1 [Internal] for Quantum: <i>cap_max_high_latency_table</i> = 1 so this whole register is reserved when Quantum	RW

28.20.9 CNCT - Congestion Notification Control Register

The CNCT register controls the congestion notification of the SwitchX device.

Table 1250 - CNCT - Congestion Notification Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1251 - CNCT - Congestion Notification Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number. If the Index value is 0xFF the operation is applied for all the ports of the device.	Index
	10:8	prio	Priority	Index
04h	31	e	CNP Enable (Default = 0) 0: disable congestion notification for this egress port 1: enable congestion notification for this egress port written only if ee is set	RW
	30	r	CNTag Remove (Default = 0) 0: keep CNTags on frames egressing to this port 1: remove CNTags from frames egressing to this port written only if re is set (Default = 0)	RW
	15	ee	Enable write operation into the e bit	WO
	14	re	Enable write operation into the r bit	WO

Note: [open: do we want to FW workaround the enable per prio issues ?](#)

28.20.10 CPCS - Congestion Point Control and Status Register

The CPCS register configures and retrieves the various parameters that control the operation of the congestion point.

Table 1252 - CPCS - Congestion Point Control and Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1252 - CPCS - Congestion Point Control and Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
cp_sample_base																															10h	
cp_min_header_octets																															14h	

Table 1253 - CPCS - Congestion Point Control and Status Register Fields

Offset	Bits	Name	Description	Access
	8	local_port	Local port number. If the Index value is 0xFF the operation will be applied for all the ports of the device	Index
00h	10:8	tclass	Traffic Class	Index
04h	31	cpqspe	Enable write operation into <i>cpqspe</i> .	WO
	30	log2_cpwe	Enable write operation into <i>cpw</i> .	WO
	29	cpsbe	Enable write operation into <i>cp_sample_base</i> .	WO
	28	cpmhoe	Enable write operation into <i>cp_min_header_octets</i> .	WO
08h	31:0	cpqsp	The set point for the queue, this is the target number of octets in the CP's queue. Default value is 26048. Values configured to this register must be 64B aligned. Maximum value for this register is 4194240. Written only if <i>cpqspe</i> is set. <i>cpqsp</i> is a shared register for all the queues in the same Tclass. Thus when setting the <i>cpqsp</i> the SwitchX device ignores the <i>local_port</i> value and applies this setting to all the <i>local_ports</i> in the this Tclass. in SwitchX this register is implemented either per SWID or pwe Tclass - once we enable SWID we might change the PRM or expose this to the user. Note that SwitchX implements Input buffering with Virtual Output queueing. The following condition might be encountered in the system: 1) Low utilization on each input buffer, might sum up to overall queue size that cross that <i>cpqsp</i> threshold thus generate CNMs though no Input buffer is really fully, nor the fabric suffers from congestion spreading. 2) Setting the threshold too high (for example three times the input buffer size) might lead to a condition where two ingress port are sending traffic to the same egress port, congestion in being build up and spread but no CNM is generated because the <i>cpqsp</i> was not crossed. System designer should find the right balance between these two conditions.	RW

Table 1253 - CPCS - Congestion Point Control and Status Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	log2_cpw	The weight (cpW) of the congestion point is equal to two to the power of this object. Thus, if this object contains a -1, cpW = 1/2. Written only if <i>log2_cpwe</i> is set. <i>cpw</i> is a shared register for all the queues in the same Tclass. Thus when setting the cpw the SwitchX device ignores the local_port value and applies this setting to all the local_ports in the this Tclass. in SwitchX this register is implemented either per SWID or pwe Tclass - once we enable SWID we might change the PRM or expose this to the user.	RW
10h	31:0	cp_sample_base	The minimum number of octets to enqueue in the CP's queue between CNM PDU transmissions. Default value 150016 Values configured to this register must be 64B aligned. Maximum value for the register is 2147483584. Written only if <i>cpsbe</i> is set <i>cp_sample_base</i> is a shared register for all the queues in the same Tclass. Thus when setting the cp_sample_base the SwitchX device ignores the local_port value and applies this setting to all the local_ports in the this Tclass. in SwitchX this register is implemented either per SWID or pwe Tclass - once we enable SWID we might change the PRM or expose this to the user.	RW
14h	31:0	cp_min_header_octets	The minimum number of octets that the CP is to return in the Encapsulated MSDU field (IEEE 802.1Qau 33.4.10) of each CNM it generates (IEEE 802.1Qau 32.9.4). Default value 0. Written only if <i>cpmhoe</i> is set. <i>cp_min_header_octets</i> is a shared register for all the queues in the same Tclass. Thus when setting the cp_min_header_octetscp_min_header_octets the SwitchX device ignores the local_port value and applies this setting to all the local_ports in the this Tclass. in SwitchX this register is implemented either per SWID or pwe Tclass - once we enable SWID we might change the PRM or expose this to the user.	RW

28.20.11CPID - Congestion Point Identifier Register

The CPID register retrieves CPID value of each congestion point.

Table 1254 - CPID - Congestion Point Identifier Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																00h
																																04h
																																08h

Table 1255 - CPID - Congestion Point Identifier Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number. if the Index value is 0xFF the operation will be applied for all the ports of the device.	Index
	10:8	prio	Priority	Index
04h	31:0	cpid_high	IEEE 802.1Qau Congestion Point Identifier.	RO
08h	31:0	cpid_low		

28.20.12CNMC - Congestion Notification Message Configuration Register

The CNMC register configures parameter associated with the CNM generation logic.

Table 1256 - CNMC - Congestion Notification Message Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h

Table 1257 - CNMC - Congestion Notification Message Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	10:8	prio	Priority	RW

28.20.13CEGCR - Congestion Elephants Global Configuration Register

*** The Elephant detection register are preliminary and subject to change ***

Table 1258 - CEGCR - Congestion Elephants Global Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch
																																10h
																																14h
																																18h
																																1Ch

Table 1259 - CEGCR - Congestion Elephants Global Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	2	f2e	Floating to Elephants If a flow is at floating state for <i>max_floating_time</i> and <i>f2e</i> is set then the flow becomes an elephant	RW
	1	sticky	Sticky Elephants 0: A flow which was detected as an elephant flow may become floating or mice 1: A flow which was detected as an elephant flow will not change state until cleared by CEER.clear	RW
	0	bits	Bits or packets 0: meter based on packets/sec 1: meter based on bits/sec	RW
04h	9:0	max_floating_time	Max floating time If a flow is at floating state for <i>max_floating_time</i> and <i>f2e</i> is set then the flow becomes an elephant Units of Sec	RW

Table 1259 - CEGCR - Congestion Elephants Global Configuration Register Fields

Offset	Bits	Name	Description	Access
10h	31:0	reference_rate	Reference Rate (RR) Flows with a rate higher than RR and a burst higher than EBS are detected as elephant flows. Flows with a rate lower than RR and a burst delta higher than MBS are detected as mice flows. Other flows are floating flows (not yet detected as elephants or mice). Units are 10^3 bits/sec or TBD Minimum allowed rate is 1.5×10^6 bps	RW
14h	19:0	elephants_bs	Elephants Burst Size (EBS) Flows with a rate higher than EDR and a burst higher than EBS are detected as elephant flows. Units are 10^6 bits or TBD [Internal] For FW: $C_i - ET > RC$	RW
18h	19:0	elephants_bs_max	Elephants Max Burst Size. Used for non-sticky mode so that an elephant flow will not gain too much credits Reserved when Spectrum-1	RW
1Ch	19:0	mice_bs	Mice Burst Size (MBS) Flows with a rate lower than EDR and a burst lower than MBS are detected as mice flows. Units are 10^6 bits or TBD [Internal] For FW: $C_i - RC < (-RT)$	RW

28.20.14CEPC - Congestion Elephants Port Configuration Register

*** The Elephant detection register are preliminary and subject to change ***

Table 1260 - CEPC - Congestion Elephants Port Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1261 - CEPC - Congestion Elephants Port Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Ingress Local Port Not supported by CPU port	Index
04h	31	en_ed	Enable Elephants detection: 0: disabled (default) 1: enabled	RW

28.20.15CEDR - Congestion Elephants Detection Register

The CEMDR returns bit induction on the detection table entries - setting bits for entries which have elephant flows. This register need to be polled by the SW.

*** *The Elephant detection register are preliminary and subject to change* ***

Table 1262 - CEDR - Congestion Elephants Port Detection Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h-0Ch
																																10h-1Ch

Table 1263 - CEDR - Congestion Elephants Port Detection Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port Not supported by CPU port [Internal] The 128bits are shared for split ports, each port will reflect only bits which have entries belonging to it	Index
10h-1Ch	128	detec- tion_table[1 28]	Detection Table Every bit represent the corresponding entry on the table 0: the entry is not an elephant (it may be mice or floating) 1: the entry is an elephant Offset 10h, bit31 is entry0	RO

28.20.16CEER - Congestion Elephants Entry Register

The CEER returns the entry from the detection table. The CEER can also clear the detection bit of the entry, prior to ACL modifications

*** The Elephant detection register are preliminary and subject to change ***

Table 1264 - CEER - Congestion Elephants Entry Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset												
clear								local_port																			detection_entry								00h									
det																																04h												
																																08h												
								egress_local_port																										0Ch										
			tunnel_type									l4_type								l3_type									vlan_type					10h										
																				inner_l3_type														14h										
hash																				lag_hash																	18h							
																				router_hash																	1Ch							
																dmac[47:32]																									20h			
dmac[31:0]																																							24h					
																smac[47:32]																									28h			
smac[31:0]																																									2Ch			
																pcp				dei	vid																		30h					
																inner_pcp				inner_dei	inner_vid																		34h					
																																		38h-3Ch										
sip																																									40h-4Ch			
dip																																												50h-5Ch
								dscp					ecn			ttl																				dont_frag	60h							

Table 1264 - CEER - Congestion Elephants Entry Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
mpls_lb3_v								mpls_label3_label_id										mpls_label3_ttl												D0h		
								mpls_label4_label_id										mpls_label4_ttl												D4h		

Table 1265 - CEER - Congestion Elephants Entry Register Fields

Offset	Bits	Name	Description	Access
00h	31	clear	Clear the elephant state of the entry. The entry becomes mice.	OP
	23:16	local_port	Local port Not supported by CPU port	Index
	6:0	detection_entry	Detection Table Entry	Index
04h	31	det	Detected The entry is detected as elephant Relevant when <i>sticky</i> is clear, see CEGCR	RO
0Ch	23:16	egress_local_port	Egress local port Reserved when Spectrum-1	RO

Table 1265 - CEER - Congestion Elephants Entry Register Fields

Offset	Bits	Name	Description	Access
10h	27:24	tunnel_type	Tunnel Type: 0: No tunnel 1: VXLAN - VXLAN, VXLAN-GPE, GENEVE 2: NVGRE 3: IP_IN_IP - no GRE 4: IP_IN_GRE_IN_IP - with or without a key [Internal] FW: gre 5: MPLS	RO
	19:16	l4_type	L4 type (L3 next header): 0: TCP 1: UDP 2: ICMP 3: IGMP 4: BTH 5: BTH_over_UDP 6: AH - IPSec Authentication Header 7: ESP - IPSec Encapsulating Security Payload 15: Unknown	RO
	11:8	l3_type	L3 type (L2 next header): 0: IPv4 1: IPv6 2: GRH 3: FiberChannel 15: Unknown [Internal] also includes ARP, FW: none, raw [Internal] FW: packet_type_l3	RO
	3:0	vlan_type	Vlan Type: 0: NO_TAG 1: PRIO_TAG 2: VLAN_TAG 3: QINQ - 2 or more VLAN tags [Internal] FW: use HW vlan_wire	RO
14h	11:8	inner_l3_type	Inner L3 type: 0: IPV4 1: other Reserved when tunnel_type is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP [Internal] FW: packet_type_l3_ipv4_or_raw_none	RO
18h	31	hash_valid		RO
	11:0	lag_hash	Reserved when hash_valid is '0'	RO
1Ch	11:0	router_hash	Reserved when hash_valid is '0'	RO
20h	15:0	dmac[47:32]	dmac	RO
24h	31:0	dmac[31:0]		
28h	15:0	smac[47:32]	smac	RO
2Ch	31:0	smac[31:0]		

Table 1265 - CEER - Congestion Elephants Entry Register Fields

Offset	Bits	Name	Description	Access
30h	15:13	pcp	Reserved when <i>vlan_type</i> is one of: NO_TAG, PRIO_TAG	RO
	12	dei	Reserved when <i>vlan_type</i> is one of: NO_TAG, PRIO_TAG	RO
	11:0	vid	VLAN ID Reserved when <i>vlan_type</i> is one of: NO_TAG, PRIO_TAG	RO
34h	15:13	inner_pcp	Reserved when <i>vlan_type</i> is not QINQ	RO
	12	inner_dei	Reserved when <i>vlan_type</i> is not QINQ	RO
	11:0	inner_vid	Inner VLAN ID Reserved when <i>vlan_type</i> is not QINQ	RO
40h-4Ch	128	sip	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH For <i>l3_type</i> of IPV4, only bits [31:0] are relevant	RO
50h-5Ch	128	dip	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH For <i>l3_type</i> of IPV4, only bits [31:0] are relevant	RO
60h	23:18	dscp	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH	RO
	17:16	ecn	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH	RO
	15:8	ttl	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH	RO
	0	dont_frag	Reserved when <i>l3_type</i> is not one of: IPv4, IPv6, GRH	RO
64h	31:16	sport	Reserved when <i>l4_type</i> is not one of: TCP, UDP	RO
	15:0	dport	Reserved when <i>l4_type</i> is not one of: TCP, UDP Reserved when <i>tunnel_type</i> is VXLAN	RO
70h	24	vni	Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE [Internal] FW: <i>tunnel_id</i>	RO
78h	15:0	inner_d-mac[47:32]	Inner DMAC Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE	RO
7Ch	31:0	inner_d-mac[31:0]		
80h	15:0	inner_s-mac[47:32]	Inner SMAC Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE	RO
84h	31:0	inner_s-mac[31:0]		
90h-9Ch	128	inner_sip	Inner source IP Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP [Internal] only 32bits relevant for Spectrum	RO
A0h-ACh	128	inner_dip	Inner destination IP Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP [Internal] only 32bits relevant for Spectrum	RO

Table 1265 - CEER - Congestion Elephants Entry Register Fields

Offset	Bits	Name	Description	Access
B0h	23:18	dscp	Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP	RO
	17:16	ecn	Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP	RO
	15:8	ttl	Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP	RO
	0	dont_frag	Reserved when <i>tunnel_type</i> is not one of: VXLAN, NVGRE, IP_IN_IP, IP_IN_GRE_IN_IP Reserved when	RO
C0h	31	mpls_bos	BoS bit was detected in the last valid label If there are many labels (more than exposed by this register, or more than allowed by parsing) then the BoS label may be '0' Reserved when <i>tunnel_type</i> is not MPLS	RO
	2:0	mpls_exp	Reserved when <i>tunnel_type</i> is not MPLS	RO
C4h	31	mpls_lb0_v	Label 0 valid Label0 is the outermost label (top of stack) Reserved when <i>tunnel_type</i> is not MPLS	RO
	27:8	mpls_label0_label_id	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb0_v</i> is '0'	RO
	7:0	mpls_label0_ttl	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb0_v</i> is '0'	RO
C8h	31	mpls_lb1_v	Reserved when <i>tunnel_type</i> is not MPLS	RO
	27:8	mpls_label1_label_id	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb1_v</i> is '0'	RO
	7:0	mpls_label1_ttl	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb1_v</i> is '0'	RO
CCh	31	mpls_lb2_v	Reserved when <i>tunnel_type</i> is not MPLS	RO
	27:8	mpls_label2_label_id	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb2_v</i> is '0'	RO
	7:0	mpls_label2_ttl	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb2_v</i> is '0'	RO
D0h	31	mpls_lb3_v	Reserved when <i>tunnel_type</i> is not MPLS	RO
	27:8	mpls_label3_label_id	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb3_v</i> is '0'	RO
	7:0	mpls_label3_label_ttl	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb3_v</i> is '0'	RO

Table 1265 - CEER - Congestion Elephants Entry Register Fields

Offset	Bits	Name	Description	Access
D4h	31	mpls_lb4_v	Reserved when <i>tunnel_type</i> is not MPLS	RO
	27:8	mpls_label4_label_id	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb4_v</i> is '0'	RO
	7:0	mpls_label4_label_ttl	Reserved when <i>tunnel_type</i> is not MPLS Reserved when <i>mpls_lb4_v</i> is '0'	RO

28.20.17CWCAM - Congestion WRED Capabilities Mask Register

The CWCAM register reports the device's supported congestion WRED features.

NOTE: For HCA devices, this register is supported only when HCA_CAP.CWCAM == 1.

Table 1266 - CWCAM - Congestion WRED Capabilities Mask Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								feature_group																access_reg_group								00h
																																04h
cw_access_reg_cap_mask																																08h-14h
																																18h-24h
feature_cap_mask																																28h-34h
																																38h-4Ch

Table 1267 - CWCAM - Congestion WRED Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	feature_group	Feature list mask index: 0: enhanced_features Values 1 - 255 are Reserved	Index
	7:0	access_reg_group	Access Register ID groups 0: RegID_0x2800_to_0x287F Values 1-255 are Reserved	Index

Table 1267 - CWCAM - Congestion WRED Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
08h-14h	128	cw_access_reg - cap_mask	Supported port's access register bitmaks. Based on <i>access_reg_group</i> index. When bit is set, The register is supported in the device. When <i>access_reg_group</i> == 0 : Bit_2: 0x2802_CWTP Bit_3: 0x2803_CWTPM Bit_31: 0x281f_CWCAM	RO
28h-34h	128	feature_cap_mask	Supported port's enhanced features. Based on <i>feature_group</i> index. When bit is set, The feature is supported in the device: Bits 0-127 are Reserved	RO

28.21 Buffer Management Registers

The table below summarizes the registers that are used for switch shared buffer management. The registers that are used for headroom shared buffer are per port and are summarized in [Section 28.19, “Network Ports Registers,” on page 1140](#).

Table 1268 - Shared Buffer Management Register Summary

Register Name	Register ID (hex)	Query	Write	InfiniBand	Ethernet	HCA	Reference
SBGCR	0xB000	v	v	-	-		Table 1269, “SBGCR - Shared Buffer Global Configuration Register Layout,” on page 1452
SBPR	0xB001	v	v	-	v	v	Table 1271, “SBPR - Shared Buffer Pools Register Layout,” on page 1453
SBCM	0xB002	v	v	-	v	v	Table 1273, “SBCM - Shared Buffer Class Management Register Layout,” on page 1455
SBPM	0xB003	v	v	-	v	v	Table 1275, “SBPM - Shared Buffer Port Management Register Layout,” on page 1458
SBMM	0xB004	v	v	-	-		Table 1277, “SBMM - Shared Buffer Multicast Management Register Layout,” on page 1460
SBME	0xB00E	v	v	-	-		Table 1279, “SBME - Shared Buffer Multicast Eligibility Register Layout,” on page 1462 [Internal] reserved when Spectrum-2
SBSR	0xB005	v	-	-	v	v	Table 1281, “SBSR - Shared Buffer Status Register Layout,” on page 1463
SBIB	0xB006	v	v	-	-		Table 1285, “SBIB - Shared Buffer Internal Buffer Register Layout,” on page 1466
SBDCC	0xB007	v	v	-	-		Table 1287, “SBDCC - Shared Buffer Drop Count CPU Register Layout,” on page 1467
SBDCM	0xB008	v	-	-	-		Table 1291, “SBDCM - Shared Buffer Drop Count Multicast Register Layout,” on page 1469
SBDRCR	0xB019	v	-	-	-		Table 1293, “SBDRCR - Shared Buffer Drop Count on Receive Register Layout,” on page 1470
SBHBR	0xB00A	v	v	-	-		Table 1295, “SBHBR - Shared Buffer Histogram Binding Register Layout,” on page 1471
SBHRR	0xB00B	v	-	-	-		Table 1300, “SBHRR - Shared Buffer Histogram Reading Register Layout,” on page 1473
SBHBR_V2	0xB011	v	v	-	-		Table 1304, “SBHBR_V2 - Shared Buffer Histogram Binding Register Layout,” on page 1475
SBHRR_V2	0xB012	v	-	-	-		Table 1306, “SBHRR_V2 - Shared Buffer Histogram Reading Register Layout,” on page 1478

Table 1268 - Shared Buffer Management Register Summary

Register Name	Register ID (hex)	Query	Write	InfiniBand	Ethernet	HCA	Reference
SBCTC	0xB00C	v	v	-	-		Table 1310, “SBCTC - Shared Buffer Congestion Threshold Configuration Register Layout,” on page 1481
SBCTR	0xB00D	v	-	-	-		Table 1312, “SBCTR - Shared Buffer Congestion Threshold Reading Register Layout,” on page 1484
SBCTS	0xB00F	v	-	-	-		Table 1314, “SBCTS - Shared Buffer Congestion Threshold Statistics Register Layout,” on page 1485
SBHPC	0xB013	v	v				Table 1316, “SBHPC - Shared Buffer Headroom Pool Configuration Register Layout,” on page 1486
SBSNT	0xB020	-	v	-	-		Table 1318, “SBSNT - Shared Buffer SNaPshot Take Register Layout,” on page 1487
SBSNS	0xB021	v	-	-	-		Table 1320, “SBSNS - Shared Buffer SNaPshot Status Register Layout,” on page 1488
SBSNTE	0xB022	v	-	-	-		Table 1322, “SBSNTE - Shared Buffer SNaPshot Trigger Enable Register Layout,” on page 1489
SBCAM	0xB01F	v	-	-	v	v	Table 1324, “SBCAM - Shared Buffer Capabilities Mask Register Layout,” on page 1490

28.21.1 SBGCR - Shared Buffer Global Configuration Register

Table 1269 - SBGCR - Shared Buffer Global Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																										cong_fp				tele_entity	00h	

Table 1270 - SBGCR - Shared Buffer Global Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	4	cong_fp	Congestion false positive For Tclass Congestion: 0: False Negative - allow trap and rules only for congested Tclass. Response may be delayed by a few uSec 1: False Positive - allow fp-trap and rules for all Tclasses, converging over time to only congested Tclasses. Provided immediate response Reserved when SwitchX/-2, Switch-IB/2, Quantum and Spectrum-2	RW
00h	1:0	tele_entity	Telemetry entity type: 0: TCLASS 1: PORT Note: change of this field is allowed only when congestion threshold is disabled on all ports and tclasses, see SBCTC Reserved when Spectrum/-2	RW

28.21.2 SBPR - Shared Buffer Pools Register

The SBPR configures and retrieves the shared buffer pools and configuration.

Table 1271 - SBPR - Shared Buffer Pools Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
desc	snap						dir																									00h
infi_size								size																								04h
																																08h
								buff_occupancy																								0Ch
clr								max_buff_occupancy																								10h
								ext_buff_occupancy																								14h

Table 1272 - SBPR - Shared Buffer Pools Register Fields

Offset	Bits	Name	Description	Access
00h	31	desc	[Advanced] Descriptor buffer For Spectrum-1: Total 84960 descriptors For NIC: Supported when <i>descriptors_cap</i> is set	Index
00h	30	snap	Read the snapshot guage 0: FREE - read free running <i>buff_occupancy</i> 1: SNAP - read snapshot of the <i>buff_occupancy</i> When SBSNR. <i>status</i> is RELEASED then the device will NACK the command	OP
	25:24	dir	Direction 0: Ingress pools 1: Egress pools	Index
	3:0	pool	Pool index Range 0 .. <i>cap_num_pool_supported</i> - 1	Index
04h	31	infi_size	size is infinite When set, the <i>size</i> field is reserved	RW
04h	23:0	size	Pool size in buffer cells Reserved when <i>infi_size</i> = 1	RW
08h	3:0	mode	Pool quota calculation mode: 0: Static 1: Dynamic	RW

Table 1272 - SBPR - Shared Buffer Pools Register Fields

Offset	Bits	Name	Description	Access
0Ch	23:0	buff_occupancy	Current buffer occupancy	RO
10h	31	clr	Clear Max Buffer Occupancy When this bit is set, the <i>max_buff_occupancy</i> field is cleared (and a new max value is tracked from the time the clear was performed)	OP
	23:0	max_buff_occupancy	Maximum value of buffer occupancy monitored. Cleared by writing to the <i>clr</i> field	RO
14h	23:0	ext_buff_occupancy	Extended current buffer occupancy Includes also all of the min_buff of the relevant PG buffer or traffic class associated to this pool Reserved when Spectrum-1, supported by Spectrum-2 and Quantum. Note: currently not supported by Switch FW <i>For NIC: Supported when ext_buff_occupancy_cap is set</i>	RO

28.21.3 SBCM - Shared Buffer Class Management Register

The SBCM register configures and retrieves the shared buffer allocation and configuration according to Port-PG, including the binding to pool and definition of the associated quota.

Table 1273 - SBCM - Shared Buffer Class Management Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
desc	snap							local_port								pnat	pg_buff																			dir	00h
																																exc	04h				
																																	08h-0Ch				
								buff_occupancy																								10h					
clr								max_buff_occupancy																								14h					
								min_buff																								18h					
infr_max								max_buff																								1Ch					
																																	20h				
																															pool		24h				

Table 1274 - SBCM - Shared Buffer Class Management Register Fields

Offset	Bits	Name	Description	Access
00h	31	desc	[Advanced] Descriptor buffer For NIC: Supported when <i>descriptors_cap</i> is set	Index
00h	30	snap	Read the snapshot guage 0: FREE - read free running <i>buff_occupancy</i> 1: SNAP - read snapshot of the <i>buff_occupancy</i> When SBSNR. <i>status</i> is RELEASED then the device will NACK the command	OP

Table 1274 - SBCM - Shared Buffer Class Management Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number	Index
	15:14	pnat	Port number access type: 0: Local_port_number 2: Host_port_number	Index
	13:8	pg_buff	PG buffer - Port PG (<i>dir</i> =ingress) / traffic class (<i>dir</i> =egress) For PG buffer: range is 0.. <i>cap_max_pg_buffers</i> - 1 For traffic class: range is 0.. <i>cap_max_tclass_sb</i> - 1 For traffic class to CPU port: range is 0.. <i>cap_max_cpu_ingress_tclass_sb</i> - 1 For traffic class to BMC port: range is 0.. <i>cap_max_bmc_tc_sb</i> - 1 [Internal] Default is pool 15, the FW does allow configuration [Internal] For Spectrum-2: - Admission for MC is also by TQ.Tclass - No pool 15 - No need for SBME Note: Spectrum family has PGs: 0..7,9	Index
	1:0	dir	Direction 0: Ingress port quota 1: Egress port quota	Index
04h	0	exc	Exclude - no accounting in the pool 0: the tclass is accounted in the pool (default) 1: the tclass is not accounted in the pool [Internal] In both cases admission is done in a regular manner, the tclass needs to check if there is space in the pool For NIC: Supported when <i>descriptors_cap</i> is set [Internal] HW: no support for PG [Internal] HW: no support for Quantum	RW
10h	23:0	buff_occupancy	Current buffer occupancy This includes cells at the pool and at the <i>min_buff</i>	RO
14h	31	clr	Clear Max Buffer Occupancy When this bit is set, the <i>max_buff_occupancy</i> field is cleared (and a new max value is tracked from the time the clear was performed)	OP
	23:0	max_buff_occupancy	Maximum value of buffer occupancy monitored. This includes cells at the pool and at the <i>min_buff</i> Cleared by writing to the <i>clr</i> field	RO

Table 1274 - SBCM - Shared Buffer Class Management Register Fields

Offset	Bits	Name	Description	Access
18h	23:0	min_buff	Minimum buffer size for the limiter The minimum buffer is not part of the pool area. Units of cells [Internal] The min_buff: - is per port.prio.dir - does not consume area in the pool - is reflected in the <i>buff_occupancy</i>	RW
1Ch	31	infi_max	Max buffer is infinite When set, the <i>max_buff</i> field is reserved [Internal] Note: user should config both data and desc [Internal] HW supports <i>infi_max</i> =1 with <i>max_buff</i> = value where the <i>infi_max</i> is used for lossless and the <i>max_buf</i> is used for lossy. Also for SBPM, SBMM.	RW
1Ch	23:0	max_buff	The maximum buffer threshold in the pool area. When the pool that is associated to the port-pg/tclass is configured to static, then <i>max_buff</i> is configured in cells When the pool that is associated to the port-pg/tclass is configured to dynamic, then <i>max_buff</i> holds the “alpha” parameter, supporting the following values: 0 : 0 i : $(1/128) * 2^{(i-1)}$, for i=1..14 0xFF: Infinity Reserved when <i>infi_max</i> = 1 [Internal] The max_buff: - is per port.prio.dir - threshold for area in the pool and in the min_buff. Example: if min_buff has 100KB and max_buff has 200KB, then total the PG can take 200KB (and not 300KB) - the area in the pool and the <i>min_buff</i> is reflected in the <i>buff_occupancy</i>	RW
24h	3:0	pool	Association of the port-priority to a pool Range 0 .. <i>cap_num_pool_supported</i> - 1	RW

28.21.4 SBPM - Shared Buffer Port Management Register

The SBPM register configures and retrieves the shared buffer allocation and configuration according to Port-Pool, including the definition of the associated quota.

Table 1275 - SBPM - Shared Buffer Port Management Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
desc	snap							local_port								pnat			pool													dir	00h
																																04h-0C0	
								buff_occupancy																								10h	
clr								max_buff_occupancy																								14h	
								min_buff																								18h	
infi_max								max_buff																								1Ch	

Table 1276 - SBPM - Shared Buffer Port Management Register Fields

Offset	Bits	Name	Description	Access
00h	31	desc	[Advanced] Descriptor buffer For NIC: Supported when <i>descriptors_cap</i> is set	Index
00h	30	snap	Read the snapshot guage 0: FREE - read free running <i>buff_occupancy</i> 1: SNAP - read snapshot of the <i>buff_occupancy</i> When SBSNR. <i>status</i> is RELEASED then the device will NACK the command	OP
00h	23:16	local_port	Local port number For Ingress: excludes CPU port For Egress: includes CPU port	Index
	15:14	pnat	Port number access type: 0: Local_port_number 2: Host_port_number	Index
	11:8	pool	The pool associated to quota counting on the <i>local_port</i> Range 0 .. <i>cap_num_pool_supported</i> - 1	Index
	1:0	dir	Direction 0: Ingress port quota 1: Egress port quota	Index

Table 1276 - SBPM - Shared Buffer Port Management Register Fields

Offset	Bits	Name	Description	Access
10h	23:0	buff_occupancy	Current buffer occupancy of the port.pool This includes cells at the pool and at the <i>min_buff</i>	RO
14h	31	clr	Clear Max Buffer Occupancy When this bit is set, <i>max_buff_occupancy</i> field is cleared (and a new max value is tracked from the time the clear was performed)	OP
	23:0	max_buff_occupancy	Maximum value of buffer occupancy monitored. This includes cells at the pool and at the <i>min_buff</i> Cleared by writing to the <i>clr</i> field	RO
18h	23:0	min_buff	Minimum buffer size The minimum buffer is not part of the pool area. Units of cells [Internal] The <i>min_buff</i> : - is per port.pool.dir - does not consume area in the pool - is reflected in the <i>buff_occupancy</i>	RW
1Ch	31	infi_max	Max buffer is infinite When set, the <i>max_buff</i> field is reserved [Internal] Note: user should config both data and desc	RW
1Ch	23:0	max_buff	The maximum buffer threshold in the pool area. When the pool is configured to static, then <i>max_buf</i> is configured in cells. When the pool is configured to dynamic, then <i>max_buff</i> holds the “alpha” parameter, supporting the following values: 0 : 0 i : $(1/128) * 2^{(i-1)}$, for i=1..14 0xFF: Infinity Reserved when <i>infi_max</i> = 1 [Internal] The <i>max_buff</i> : - is per port.pool.dir - threshold for area in the pool - the area in the pool and the <i>min_buff</i> is reflected in the <i>buff_occupancy</i>	RW

28.21.5 SBMM - Shared Buffer Multicast Management Register

The SBMM register configures and retrieves the shared buffer allocation and configuration for MC packets according to Switch-Priority, including the binding to pool and definition of the associated quota.

Table 1277 - SBMM - Shared Buffer Multicast Management Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
	snap																				prio											00h
																															exc	04h
																																08h-0Ch
																																10h
																																14h
																																18h
																																1Ch
																																20h
																																24h

Table 1278 - SBMM - Shared Buffer Multicast Management Register Fields

Offset	Bits	Name	Description	Access
00h	30	snap	Read the snapshot guage 0: FREE - read free running <i>buff_occupancy</i> 1: SNAP - read snapshot of the <i>buff_occupancy</i> When SBSNR. <i>status</i> is RELEASED then the device will NACK the command Reserved when Spectrum-1	OP
00h	11:8	prio	Switch Priority	Index
04h	0	exc	[Internal] Not supported, placeholder for future Exclude - no accounting in the pool	RW
10h	23:0	buff_occupancy	Current buffer occupancy	RO

Table 1278 - SBMM - Shared Buffer Multicast Management Register Fields

Offset	Bits	Name	Description	Access
14h	31	clr	Clear Max Buffer Occupancy When this bit is set, the <i>max_buff_occupancy</i> field is cleared (and a new max value is tracked from the time the clear was performed)	OP
	23:0	max_buff_occupancy	Maximum value of buffer occupancy monitored. Cleared by writing to the <i>clr</i> field	RO
18h	23:0	min_buff	Minimum buffer size for the limiter, in cells	RW
1Ch	31	infi_max	Max buffer is infinite When set, the <i>max_buff</i> field is reserved	RW
1Ch	23:0	max_buff	When the pool associated to the port-pg/tclass is configured to static, Maximum buffer size for the limiter configured in cells. When the pool associated to the port-pg/tclass is configured to dynamic, the <i>max_buff</i> holds the “alpha” parameter, supporting the following values: 0 : 0 i : $(1/128) * 2^{(i-1)}$, for i=1..14 0xFF: Infinity Reserved when <i>infi_max</i> = 1	RW
24h	3:0	pool	Association of the port-priority to a pool	RW

28.21.6 SBME - Shared Buffer Multicast Eligibility Register

The SBME register configures the pool for eligibility for multicast packets. For good practice, this should be reconfigured if mode was MC aware/unaware mode was changed, see QTCTM.

For Spectrum-2 the eligibility is by egress_port.Tclass.pool (SBCM), so this register is not needed

[Internal] MC-Aware mode: when there are allot of MC (including BC/Flood) packets then it is recommended to work in MC-aware mode so that MC packets will have dedicated TClasses which are not limited by pools of the UC TClasses. For MC-aware it is not recommended to configure the SBME since MC packets may be counted multi-times and thus UC pools may appear as full. Therefore, FW default is to use pool 15

[Internal] When there are not allot of MC (including BC/Flood) packets then one may use the MC un-aware mode in order to have 16 TClasses. However, for Spectrum a packet will decide for shared buffer eligibility according to the egress_port.pool (SBME) but will consume space in the egress_port.Tclass.pool (SBCM). Thus it is recommended to point the egress_port pool to one of the pools used by the egress_port.

[Internal] This register configures for both MC-aware and MC-unaware modes.

Table 1279 - SBME - Shared Buffer Multicast Eligibility Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
desc								local_port																									00h
																																04h-20h	
																												pool			24h		

Table 1280 - SBME - Shared Buffer Multicast Eligibility Register Fields

Offset	Bits	Name	Description	Access
00h	31	desc	[Advanced] Descriptor buffer	Index
00h	23:16	local_port	Local port number for egress ports Includes CPU port	Index
24h	3:0	pool	Association of the port-priority to a pool	RW

28.21.7 SBSR - Shared Buffer Status Register

The SBSR register retrieves the shared buffer occupancy according to Port-Pool. Note that this register enables reading a large amount of data. It is the user's responsibility to limit the amount of data to ensure the response can match the maximum transfer unit. In case the response exceed the maximum transport unit, it will be truncated with no special notice however all requested entries will be cleared if *clr* is set.

Table 1281 - SBSR - Shared Buffer Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
clr	snat															pnat															ports	00h		
desc																																04h		
																																08h		
																																0Ch		
ingress_port_mask																																10h-2Ch		
																								pg_buff_mask										30h
egress_port_mask																																34h-50h		
tclass_mask																																54h-58h		
shared_buffer_status[0](Table 1284, “Shared Buffer Status Fields,” on page 1465)																																5Ch-60h		
shared_buffer_status[1](Table 1284, “Shared Buffer Status Fields,” on page 1465)																																64h-68h		
...																																...		
shared_buffer_status[19](Table 1284, “Shared Buffer Status Fields,” on page 1465)																																F4h-F8h		

Table 1282 - SBSR - Shared Buffer Status Register Fields

Offset	Bits	Name	Description	Access
00h	31	clr	Clear Max Buffer Occupancy When this bit is set, the <i>max_buff_occupancy</i> field is cleared (and a new max value is tracked from the time the clear was performed)	OP

Table 1282 - SBSR - Shared Buffer Status Register Fields

Offset	Bits	Name	Description	Access
00h	30	snap	Read the snapshot gauge 0: FREE - read free running <i>buff_occupancy</i> 1: SNAP - read snapshot of the <i>buff_occupancy</i> When SBSNR. <i>status</i> is RELEASED then the device will NACK the command	Index
	15:14	pnat	Port number access type: 0: Local_port_number 2: Host_port_number	Index
	0	ports	Ports: 0: read per pg_buff and/or tclass 1: ports, read per port [Internal] As for Feb. 2018 it is not yet implemented by FW. This is needed for Spc-2 but will also work on Spc-1	Index
04h	31	desc	[Advanced] Descriptor buffer For NIC: Supported when <i>descriptors_cap</i> is set	Index
10h-2Ch	256	ingress_port_mask	Bit vector for all ingress network port. Indicates which of the ports (for which the relevant bit is set) are affected by the set operation. Configuration of any other port does not change.	Index
30h	9:0	pg_buff_mask	Bit vector for all priority groups. Indicates which of the priorities (for which the relevant bit is set) are affected by the set operation. Configuration of any other priority does not change. Range is 0.. <i>cap_max_pg_buffers</i> - 1 Reserved when <i>ports</i> = 1	Index
34h-50h	256	egress_port_mask	Bit vector for all egress network port. Indicates which of the ports (for which the relevant bit is set) are affected by the set operation. Configuration of any other port does not change.	Index
54h-58h	64	tclass_mask	Bit vector for all traffic classes. Indicates which of the traffic classes (for which the relevant bit is set) are affected by the set operation. Configuration of any other priority does not change. Range is 0.. <i>cap_max_tclass</i> - 1 For traffic class to CPU port: range is 0.. <i>cap_max_cpu_ingress_tclass</i> - 1 Reserved when <i>ports</i> = 1	Index

Table 1282 - SBSR - Shared Buffer Status Register Fields

Offset	Bits	Name	Description	Access
5Ch-F8h	64	shared_buffer_status[20](Table 1284, “Shared Buffer Status Fields,” on page 1465)	Shared Buffer Properties, see Table 1284, “Shared Buffer Status Fields,” on page 1465 This field is a collection of all the statuses of the requested quotas, while $i=0..\min\{\text{requested quotas}, \text{cap\_sbsr\_stat\_size}-1\}$ The i th status refers a quota as follows; The occurrences of the masked ingress quotas are written before the masked egress quotas. All the occurrences associated to port are written together from the smallest masked port index. In each port, the occurrences of the masked priority groups/ traffic classes are written from the lowest to the highest.	RO

Table 1283 - Shared Buffer Status Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1284 - Shared Buffer Status Fields

Offset	Bits	Name	Description	Access
00h	23:0	buff_occupancy	Current buffer occupancy	RO
04h	23:0	max_buff_occupancy	Maximum value of buffer occupancy monitored. Cleared by writing to the <i>clr</i> field	RO

28.21.8 SBIB - Shared Buffer Internal Buffer Register

The SBIB register configures per port buffers for internal use. The internal buffers consume memory on the port buffers (note that the port buffers are used also by PBMC).

Table 1285 - SBIB - Shared Buffer Internal Buffer Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type								local_port																int_buffer_index								00h
status																																04h
								buff_size																								08h
																																0Ch

Table 1286 - SBIB - Shared Buffer Internal Buffer Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	type	0: Internal buffer allocated on the ports Other values not supported by Spectrum This is a placeholder for future functions	Index
00h	23:16	local_port	Local port number Not supported for CPU port Not supported for Router port	Index
	7:0	int_buffer_index	Internal Buffer Index Spectrum supports only 1 index per port with value of 0	Index
04h	31:28	status	Status: 0: the internal buffer status is not yet at the size of <i>buff_size</i> 1: the internal buffer status is equal to <i>buff_size</i>	RO
08h	23:0	buff_size	Buffer size Units represented in cells Allowed range is 0 to (<i>cap_max_headroom_size - 1</i>) Default is 0	RW

28.21.9 SBDCC - Shared Buffer Drop Count CPU

The SBDCC register allows reading drop counters of packets to CPU.

Table 1287 - SBDCC - Shared Buffer Drop Count CPU Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
clr																																00h
																																04h-0Ch
no_buffer_discard_cpu_tclass[0]																															10h-14h	
no_buffer_discard_cpu_tclass[1]																															18h-1Ch	
...																															...	
no_buffer_discard_cpu_tclass[63]																															208h-20Ch	

Table 1288 - SBDCC - Shared Buffer Drop Count CPU Register Fields

Offset	Bits	Name	Description	Access
00h	31	clr	Clear counters	OP
10h-20Ch	64	no_buffer_discard_cpu_tclass[64]	Count the number of unicast packets to CPU dropped due to lack of shared buffer resources For tclass i Range is 0.. <i>cap_max_cpu_ingress_tclass</i> -1 See Table 1289, “SBDCC - counter Layout,” on page 1467	RO

Table 1289 - SBDCC - counter Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
count_high																															00h	
count_low																															04h	

Table 1290 - SBDCC - counter Fields

Offset	Bits	Name	Description	Access
00h	31:0	count_high	Count the number of unicast packets to CPU dropped due to lack of shared buffer resources	RO
04h	31:0	count_low		

28.21.10SBDCM - Shared Buffer Drop Count Multicast

The SBDCM register allows binding drop counters of multicast packets per switch priority. Note that in PPCNT there is a drop counter of multicast packets per Tx port.

Table 1291 - SBDCM - Shared Buffer Drop Count Multicast Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
counter_set_type								counter_index																								10h

Table 1292 - SBDCM - Shared Buffer Drop Count Multicast Register Fields

Offset	Bits	Name	Description	Access
10h	31:24	counter_set_type	Counter Set Type for flow counters. default: type=0 (disabled) Not supported by SwitchX/-2 and Spectrum-1	RW
	23:0	counter_index	Counter Index for flow counters, counting no buffer discard per switch priority.	RW

28.21.11SBDCR - Shared Buffer Drop Count on Receive

The SBDCR register allows binding drop counters due to receive errors. Note that when there is a receive buffer drop both the egress drop counters and receive drop counts will both count.

The counters can be read per PG by MGPC or by PPCNT.*rx_no_buffer_discard_uc*

Table 1293 - SBDCR - Shared Buffer Drop Count on Receive Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
counter_set_type								counter_index																								10h

Table 1294 - SBDCR - Shared Buffer Drop Count on Receive Register Fields

Offset	Bits	Name	Description	Access
10h	31:24	counter_set_type	Counter Set Type for flow counters.	RW
	23:0	counter_index	Counter Index for flow counters, counting no buffer discard per switch priority. Reserves a counter range of num_ports*num_PGs: PG 0..7 - PGs PG 8 - Control PG PG 9 - Mirror PG PG 10 - egress-traps, reserved when Spectrum-2 [Internal] used by Quantum	RW

28.21.12SBHBR - Shared Buffer Histogram Binding Register

The SBHBR binds queue histogram agents.

Table 1295 - SBHBR - Shared Buffer Histogram Binding Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode								local_port																								00h
																																04h
																																08h
																hist_type																0Ch
																hist_parameters																10h
																hist_min_value																14h
																hist_max_value																18h
																																sample_time

Table 1296 - SBHBR - Shared Buffer Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	opcode	Opcode: 0: Bind a <i>hist_id</i> 1: Unbind a <i>hist_id</i> (default) <i>[Internal]</i> when binding an already bound <i>hist_id</i> , the histograms continues, so it is suggested to clear the histogram after the re-binding	RW
	23:16	local_port	Local port number Not supported for CPU port	RW
04h	7:0	hist_id	Histogram id to bind from 0 to <i>cap_max_queue_hist_id</i> - 1	Index
08h	15:0	hist_type	0x1000: Queue_Depth_Traffic_Class See Table 1297, “SBHBR - Histograms Hist-type for queue histograms,” on page 1472	RW
0Ch	31:0	hist_parameters	According to <i>hist_type</i> See Table 1297, “SBHBR - Histograms Hist-type for queue histograms,” on page 1472	RW
10h	31:0	hist_min_value	Min value for histogram main range Units according to the units of the <i>hist_type</i>	RW
14h	31:0	hist_max_value	Max value for histogram main range Note: for Spectrum device, <i>hist_group</i> =1 this rule must be meet: <i>hist_max_value</i> = <i>hist_min_value</i> + 2 ⁿ , n >= 3 Units according to the units of the <i>hist_type</i>	RW

Table 1296 - SBHBR - Shared Buffer Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
18h	5:0	sample_time	Sample time interval. Allowed range: $time = 2^{\text{sample\_time}} * 128\text{nSec}$ <i>sample_time</i> range is 0.. <i>cap_max_sample_time</i> -1 Note: for Spectrum the <i>sample_time</i> is global for all ports and all tclass For Spectrum-2 the sample_time is per port.tclass. [Internal] also for Switch-IB/2 and Quantum it is per port.VL [Internal] up to 2^{11} for Switch-IB/2 and Spectrum, TBD for Quantum and Spectrum-2	RW

Table 1297 - SBHBR - Histograms Hist-type for queue histograms

Entity ID	Entity Name	Entity parameters	Units	sample_time	SwitchX-2	Switch-IB/2	Quantum	Spectrum-1	Spectrum-2
1000h	Queue_Depth_Traffic_Class	Prio_Tclass, see Table 1298, “SBHBR - Hist-Parameters for Port-Tclass Layout,” on page 1472	Cells	v	-	-	-	v	v
1004h	Queue_Depth_Port	NA This is per tx port	Cells	v	-	-	v	-	-

28.21.12.1 Hist-Parameters for Port-Tclass

Table 1298 - SBHBR - Hist-Parameters for Port-Tclass Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																									tclass							00h

Table 1299 - SBHBR - Hist-Parameters for Port-Tclass Fields

Offset	Bits	Name	Description	Access
00h	7:0	tclass	Traffic Class Range: 0.. <i>cap_max_tclass</i> -1	RW

28.21.13 SBHRR - Shared Buffer Histogram Reading Register

The SBHRR allows reading a histogram agent.

Table 1300 - SBHRR - Shared Buffer Histogram Reading Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
clr																																00h	
																									hist_id							04h	
																																	08h-0Ch
bin[0]																															10h-14h		
bin[1]																															18h-1Ch		
...																															...		
bin[9]																															58h-5Ch		

Table 1301 - SBHRR - Port Histogram Reading Register Fields

Offset	Bits	Name	Description	Access
00h	31	clr	Clear: 0: Read the data in the histogram agent 1: Read and clear the data in the histogram agent	OP
04h	7:0	hist_id	see SBHBR	Index
10h-5Ch	64	bin[10]	Value of bin Units according to the units of the <i>hist_type</i> which is used for the <i>hist_id</i> Max bins is according to <i>cap_hist_num_bins</i> Bin is saturated at $(2^{\text{cap\_hist\_bin\_size\_type1}} - 1)$. When any bin is saturated all bins stop incrementing. See Table 1302, "SBHRR - Bin Layout," on page 1473	RO

Table 1302 - SBHRR - Bin Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bin_high																															00h	
bin_low																															04h	

Table 1303 - SBHRR - Bin Fields

Offset	Bits	Name	Description	Access
00h	31:0	bin_high		RO
04h	31:0	bin_low		RO

28.21.14SBHBR_V2 - Shared Buffer Histogram Binding Version 2 Register

The SBHBR_V2 configures the Rx.port.PG and Tx.port.Tclass pre-allocated queue depth histograms and the Tx.port.Tclass pre-allocated queue latency histograms.

Reserved when Spectrum-1.

[Internal] Spectrum-2,3,4:

For Queue_Depth: Ingress (not from cpu) 9 PGs, egress (also to CPU) 17 tclasses

For Latency: egress (not to/from CPU), 16 tclasses

[Internal] the SBHBR_V2 and SBHRR_V2 are supported by Quantum FW (for debug only???)

Table 1304 - SBHBR_V2 - Shared Buffer Histogram Binding Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port									pg_buff									dir				00h		
																															en	04h
	mode															hist_type																08h
																																0Ch
hist_min_value																																10h
hist_max_value																																14h
																											sample_time				18h	

Table 1305 - SBHBR_V2 - Shared Buffer Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number. For Ingress: excludes CPU port and Router port For Egress: excludes IP Router For Egress hist_type of Queue_Latency: excludes also CPU port	Index
	13:8	pg_buff	PG buffer - Port PG (<i>dir</i> =ingress) / traffic class (<i>dir</i> =egress) For PG buffer: range is 0.. <i>cap_max_pg_buffers</i> - 1 For traffic class, hist_type of Queue_Depth_Traffic_Class: range is 0.. <i>cap_max_tclass</i> - 1 For traffic class, hist_type of Queue_Latency: range is 0.. <i>cap_max_tclass_data</i> - 1 For traffic class to CPU port: range is 0.. <i>cap_max_cpu_ingress_tclass</i> - 1 For Spectrum: no admission on Tx.Tclass, so when traffic class is in MC aware mode then the traffic classes which are MC aware must not be configured, but can be read. Spectrum-2 has admission on Tx.Tclass so all traffic classes can be configured also for MC aware mode. [Internal] Default is pool 15, the FW does allow configuration [Internal] For Spectrum-2: - Admission for MC is also by TQ.Tclass - No pool 15 - No need for SBME	Index
	1:0	dir	Direction 0: Ingress port quota 1: Egress port quota Reserved when <i>hist_type</i> = Queue_Latency and this field is treated as '1' Egress port quota [Internal] when RLM (Remote Latency Measurement for debug) mode then ingress/egress is by MORLM.dir	Index
04h	0	en	When <i>hist_type</i> = Queue_Depth_Traffic_Class: Enable the histogram When <i>hist_type</i> = Queue_Latency: This is a WO for clearing the histograms and the counters	RW
08h	30:28	mode	mode 0: Linear histogram 1: Exponential histogram	RW
	15:0	hist_type	0x1000: Queue_Depth_Traffic_Class 0x1002: Queue_Latency See Table 1297, "SBHBR - Histograms Hist-type for queue histograms," on page 1472	Index
10h	31:0	hist_min_value	Min value for histogram main range Units according to the units of the <i>hist_type</i>	RW

Table 1305 - SBHBR_V2 - Shared Buffer Histogram Binding Register Fields

Offset	Bits	Name	Description	Access
14h	31:0	hist_max_value	Max value for histogram main range When <i>mode</i>=0 (linear), then this rule must be meet: $hist\_max\_value = hist\_min\_value + 2^n, n \geq 3$ When <i>mode</i>=1 (exponential), then this rule must be meet: $hist\_max\_value = hist\_min\_value + 255 * 2^n, n \geq 0$ Units according to the units of the <i>hist_type</i>	RW
18h	5:0	sample_time	Sample time interval. Allowed range: $time = 2^{sample\_time} * 128nSec$ <i>sample_time</i> range is 0..<i>cap_max_sample_time</i>-1 For Spectrum-1 the <i>sample_time</i> is global for all ports and all tclass For Spectrum-2 the <i>sample_time</i> is per port.tclass. [Internal] also for Switch-IB/2 and Quantum it is per port.VL [Internal] up to 2¹¹ for Switch-IB/2 and Spectrum, TBD for Quantum and Spectrum-2 Reserved when <i>hist_type</i> = Queue_Latency	RW

28.21.15SBHRR_V2 - Shared Buffer Histogram Reading Version 2 Register

The SBHRR_V2 allows reading the Rx.port.PG and Tx.port.Tclass pre-allocated histograms.
Reserved when Spectrum-1.

Table 1306 - SBHRR_V2 - Shared Buffer Histogram Reading Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
clr								local_port										pg_buff																		dir	00h
																																04h					
																hist_type																08h					
																																0Ch					
min_sampled_high																	10h																				
min_sampled_low																	14h																				
max_sampled_high																	18h																				
max_sampled_low																	1Ch																				
																																20h-3Ch					
bin[0]																	40h-44h																				
bin[1]																	48h-4Ch																				
...																	...																				
bin[9]																	88h-8Ch																				

Table 1307 - SBHRR_V2 - Port Histogram Reading Register Fields

Offset	Bits	Name	Description	Access
00h	31	clr	Clear: 0: Read the data in the histogram agent 1: Read and clear the data in the histogram agent	OP

Table 1307 - SBHRR_V2 - Port Histogram Reading Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number. For Ingress: excludes CPU port For Egress: includes CPU port	Index
	13:8	pg_buff	PG buffer - Port PG (<i>dir</i> =ingress) / traffic class (<i>dir</i> =egress) For PG buffer: range is 0.. <i>cap_max_pg_buffers</i> - 1 For traffic class: range is 0.. <i>cap_max_tclass</i> - 1 For traffic class to CPU port: range is 0.. <i>cap_max_cpu_ingress_tclass</i> - 1	Index
	1:0	dir	Direction 0: Ingress port quota 1: Egress port quota Reserved when <i>hist_type</i> = Queue_Latency and this field is treated as '1' Egress port quota	Index
08h	15:0	hist_type	0x1000: Queue_Depth_Traffic_Class 0x1002: Queue_Latency See Table 1297, "SBHBR - Histograms Hist-type for queue histograms," on page 1472	Index
10h	31:0	min_sampled_high	min_sampled Cleared when <i>clr</i> = 1, cleared to 0xffff ffff ffff ffff Units according to the units of the <i>hist_type</i> which is used for the <i>hist_id</i> Reserved when <i>hist_type</i> = Queue_Depth_Traffic_Class	RO
14h	31:0	min_sampled_low		
18h	31:0	max_sampled_high	max_sampled Cleared when <i>clr</i> = 1, cleared to 0 Units according to the units of the <i>hist_type</i> which is used for the <i>hist_id</i> Reserved when <i>hist_type</i> = Queue_Depth_Traffic_Class	RO
1Ch	31:0	max_sampled_low		
40h-8Ch	64	bin[10]	Value of bin Units according to the units of the <i>hist_type</i> which is used for the <i>hist_id</i> Max bins is according to <i>cap_hist_num_bins</i> . Bin is saturated at ($2^{\wedge}cap\_hist\_bin\_size\_type1-1$). When any bin is saturated all bins stop incrementing. See Table 1308, "SBHRR_V2 - Bin Layout," on page 1479	RO

Table 1308 - SBHRR_V2 - Bin Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
bin_high																															00h	
bin_low																															04h	

Table 1309 - SBHRR_V2 - Bin Fields

Offset	Bits	Name	Description	Access
00h	31:0	bin_high		RO
04h	31:0	bin_low		RO

28.21.16SBCTC - Shared Buffer Congestion Threshold Configuration Register

The SBCTC sets a congestion threshold per transmit port for reporting congestion event. The egress port has a single threshold. Each TClass can be enabled. A non enabled TClass has an “infinite” threshold.

For Quantum and for Spectrum-2 the congestion state is used for:

1. congestion traps
2. congestion mirror
3. shared buffer snapshot

For IB devices the TClass refers to VLs.

Table 1310 - SBCTC - Shared Buffer Congestion Threshold Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
						dir_in g		local_port																									mode	00h
en config																															event	04h		
tclass_en_high																																08h		
tclass_en_low																																0Ch		
									thr_max																		10h							
									thr_min																		14h							
																																18h-1Ch		
tclass_fix_high																																20h		
tclass_fix_low																																24h		

Table 1311 - SBCTC - Shared Buffer Congestion Threshold Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	25:24	dir_ing	Direction 0: Egress Tclass 1: Ingress PG Reserved when Spectrum-1 [Internal] the SBCTC is also used for congestion mirroring for Quantum and Spectrum-2	Index
00h	23:16	local_port	Local port number for egress sport Not supported for CPU port	Index

Table 1311 - SBCTC - Shared Buffer Congestion Threshold Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	0	mode	Mode of threshold: 0: Fixed 1: Percentage Reserved when Spectrum-1 and Quantum [Internal] HW config is per tclass/pg [Internal] Quantum HW supports, but if no FW of shared-buf then no need percentage	RW
04h	31	en_config	Enable configuration of all fields except <i>event</i> field	WO
	1:0	event	Event generation on crossing thr max for this port: 0: NO_EVENT - Do not generate event 1: EVENTS - Generate events for crossing up and down Not configured when <i>en_config</i> = 0	RW
08h	31:0	tclass_en_high	TClass/PG Trap enable (for IB switches this is VL enable) Each bit represents corresponding tclass 0: disable (default) 1: enable Valid bits when SBGCR. <i>tele_entity</i> = PORT: bit0 (all tclasses), else: Valid bits for Ethernet devices: 0.. <i>cap_max_tclass_data</i> - 1	RW
0Ch	31:0	tclass_en_low		
10h	23:0	thr_max	Threshold max When an egress port TClass crosses the threshold (cross up or down) then the HW will emit an event of SB_CONG_TX_PORT and the status can be read by SBCTR For Quantum and Spectrum-2: When an egress port TClass/PG crosses the threshold then congestion mirror is enabled When congestion mode is Fixed: - Cells. Valid inputs are 1.. <i>cap_total_buffer_size/cap_cell_size</i> When congestion mode is Percentage: - Percentage of the quota of local_port,traffic_class. Range is 0,..100, units of integer percentage. Written only if <i>en_config</i> = 1 [Internal] For Switch-IB/2, the adaptive routing high threshold uses the same threshold as configured by SBCTC	RW

Table 1311 - SBCTC - Shared Buffer Congestion Threshold Configuration Register Fields

Offset	Bits	Name	Description	Access
14h	23:0	thr_min	[Internal] Quantum and Spectrum-2 have HW support, currently no FW support (Oct. 2017) Threshold min When an egress port tclass crosses below the threshold (cross up or down) then the congestion state is change to set/clear. The status can be read by SBCTR When congestion mode is Fixed: - Cells. Valid inputs are 1..cap_total_buffer_size/cap_cell_size When congestion mode is Percentage: - Percentage of the quota of local_port,traffic_class. Range is 0..100, units of integer percentage. Written only if <i>en_config</i> = 1 Value of 0 means that the hysteresis is disabled [Internal] FW will set thr_min=thr_max	RW
20h	31:0	tclass_fix_high	[Internal] Quantum and Spectrum-2 have HW support, currently no FW support (Oct. 2017) Mode of thresholds Each bit represents corresponding TClass/PG/VL 0: Fixed 1: Percentage	RW
24h	31:0	tclass_fix_low		

28.21.17SBCTR - Shared Buffer Congestion Threshold Reading Register

The SBCTR allows reading per port which tclasses or PGs are congested above the congestion threshold.

Access to this register can also be done via an event with trap_id SB_CONG_TX_PORT.

Table 1312 - SBCTR - Shared Buffer Congestion Threshold Reading Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ievent																																00h
																																04h
																																08h
																																0Ch

Table 1313 - SBCTR - Shared Buffer Congestion Threshold Reading Register Fields

Offset	Bits	Name	Description	Access
00h	31	ievent	Invoke event 0: Read only 1: Does both reading and invoke an event of type SB_CONG_TX_PORT	OP
	23:16	local_port	Local port number Not supported for CPU port	Index
	9:8	dir_ing	Direction 0: Egress Tclass 1: Ingress PG Reserved when Spectrum-1	Index
	4	fp	False Positive When a trap: this is a false positive trap Reserved when Switch-IB/2, Quantum and Spectrum-2	RO
	1:0	entity	Entity type: 0: TCLASS 1: PORT This field represents the value configured by SBGCR. <i>tele_entity</i>	RO
08h	31:0	tclass_vector_high	Tclass vector Each bit represents corresponding tclass 0: Tclass is below the congestion threshold 1: Tclass is above the congestion threshold Note that a TClass which is disabled from threshold will always be below the congested threshold. See SBCTC Valid bits when <i>entity</i> = PORT: bit0 (all tclasses), else: Valid bits for Ethernet devices: 0.. <i>cap_max_tclass_data</i> - 1	RO
0Ch	31:0	tclass_vector_low		

28.21.18SBCTS - Shared Buffer Congestion Threshold Statistics Register

The SBCTS allows reading congestion threshold statistics.

Table 1314 - SBCTS - Shared Buffer Congestion Threshold Statistics Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
clear																																00h
																																04h-0Ch
																																10h

Table 1315 - SBCTS - Shared Buffer Congestion Threshold Statistics Register Fields

Offset	Bits	Name	Description	Access
00h	31	clear	Clear counter 0: Do not clear 1: Clear the counter	OP
10h	23:0	trap_over-flow_counter	Trap Overflow Counter Counts how many congestion traps (SB_CONG_TX_PORT) were not invoked by the device due to trap overflow Counter is sticky when reaching 2 ²⁴ -1	RO

28.21.19SBHPC - Shared Buffer Headroom Pool Configuration Register

The SBHPC configures and allows reading status of the SHP (Shared Headroom Pool).

Table 1316 - SBHPC - Shared Buffer Headroom Pool Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
																																10h
																																14h-1Ch
																																20h

Table 1317 - SBHPC - Shared Buffer Headroom Pool Configuration Register Fields

Offset	Bits	Name	Description	Access
10h	23:0	max_buff	Maximum SHP (Shared Headroom Pool) size. Default:0 When <i>max_buff</i> >0 then ports which have <i>PBMC.shp_max_loan</i> >0 may use the SHP	RW
20h	23:0	buff_occupancy	Current SHP occupancy [Internal] HW has no watermark, only the gauge	RO

28.21.20SBSNT - Shared Buffer Snapshot Take Register

The SBSNT takes and releases snapshots.

Table 1318 - SBSNT - Shared Buffer Snapshot Take Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															take	00h

Table 1319 - SBSNT - Shared Buffer Snapshot Take Register Fields

Offset	Bits	Name	Description	Access
00h	1:0	take	Take snapshot: 0: RELEASE - release snapshot 1: TAKE_AND_TRAP - take snapshot of the guages and invoke trap SB_SNAPSHOT 2: TAKE_NO_TRAP - take snapshot of the guages without invoking a trap When snapshot status is LOCKED then the TAKE will be ignored When snapshot status is RELEASED then the RELEASE will be ignored See SBSNS. <i>status</i>	OP

28.21.21 SBSNS - Shared Buffer Snapshot Status Register

The SBSNS provides snapshot status and details about last snapshot.

Table 1320 - SBSNS - Shared Buffer Snapshot Status Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															status	00h
																															trigger_id	04h
																															trigger_parameters	08h
																																0Ch
																															time_high	10h
																															time_low	14h

Table 1321 - SBSNS - Shared Buffer Snapshot Status Register Fields

Offset	Bits	Name	Description	Access
00h	0	status	Snapshot status: 0: RELEASED - snapshot is non-active 1: LOCKED - snapshot is active [Internal] HW time from SBSNT.RELEASE command to RELEASED state is up to 10uSec. Thus when status is RELEASED the next take may be delayed by up to 10uSec (if the TAKE was given immediately when status become RELEASED)	RO
04h	3:0	trigger_id	Tells which trigger id has trigger the snapshot.	RO
08h	15:0	trigger_parameters	Trigger parameters.	RO
10h	5:0	time_high	Snapshot time:	RO
14h	31:0	time_low	The time when the last snapshot was taken as free_running - clock [Internal] HW has 38bit glc_free_running_clock At init FW needs to measure the delta of frc and gfr Then FW adds/subs the delta from gfr to frc	RO

28.21.22SBSNTE - Shared Buffer SNApshot Trigger Enable

The register is used to configure the shared buffer Snapshot trigger enable for different reasons.

Table 1322 - SBSNTE - Shared Buffer SNApshot Trigger Enable Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch

Table 1323 - SBSNTE - Shared Buffer SNApshot Trigger Enable Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local Port	Index
04h	3:0	type	Type: 0x20: WRED, egress, no CPU port 0x31: Shared-buffer-per-Tclass, egress 0x40: ING-CONG, ingress 0x50: EGR-CONG, egress [Internal] for future: 0x60: ECN, egress, no CPU port [Internal] for future: 0x70: Latency, egress, no CPU port	Index
08h	31:0	tclass_en_high	TClass/PG snapshot trigger enable (for IB switches this is VL enable) Each bit represents corresponding tclass 0: disable (default) 1: enable Valid bits for Ethernet devices: 0.. <i>cap_max_tclass_data</i> - 1	RW
0Ch	31:0	tclass_en_low		

28.21.23SBCAM - Shared Buffer Capabilities Mask Register

The SBCAM register reports the device supported Shared Buffer registers and features.

NOTE: For HCA devices, this register is supported only when `HCA_CAP.SBCAM == 1`.

Table 1324 - SBCAM - Shared Buffer Capabilities Mask Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-14h
																																18h-24h
																																28h-34h
																																38h
																																3Ch
																																40h
																																44h
																																48h

Table 1325 - SBCAM - Shared Buffer Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	feature_group	Feature list mask index 0: Enhanced features	Index
	7:0	access_reg_group	Access Register ID groups 0: First 128 REG_ID	Index
08h-14h	128	sb_access_reg_cap_mask	Supported Shared Buffer access register bitmasks. Based on access_reg_group index. When bit is set, the register is supported in the device. For example: Bit 1: SBPR_(0xB001) Bit 2: SBCM_(0xB002) Bit 3: SBPM_(0xB003) Bit 5: SBSR_(0xB005)	RO

Table 1325 - SBCAM - Shared Buffer Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
28h-34h	128	sb_feature_cap_mask	Supported Shared Buffer enhanced features based on <i>feature_group</i> index. When bit is set, the feature is supported in the device Bit 0: <i>descriptors_cap</i> - (desc) is supported Bit 1: <i>ext_buff_occupancy_cap</i> - is supported Bit 2: <i>exclude_cap</i> - bit is supported Bit 3: <i>sb_own_res</i> - user can configure ePort per tc shared buffer configuration Bit 4: <i>sb_global_res</i> - user can configure ePort, iPort and iPort per pg shared buffer configuration	RO
3Ch	31:16	cap_total_descriptors	Total amount of descriptors in the device - 16 bits	RO
40h	31:0	cap_total_buffer_size	Total buffer size in the device. Units of Bytes.	RO
44h	31:16	cap_cell_size	Buffer cell size (Bytes)	RO
	15:8	cap_max_pg_buffers	Maximum number of port group buffers	RO
	7:0	cap_num_pool_supported	Number of shared buffer pools supported	RO
48h	23:16	cap_sbsr_stat_size	SBSR register number of responses	RO
	15:8	cap_max_tclass_sb	Maximum number of tclasses	RO
	7:0	cap_max_cpu_ingress_tclass_sb	Maximum number of tclasses for bmc port	RO

28.22 Management and Peripherals Control Registers

The following section lists the various registers used to configure/query the HCA.

[Table 1326](#) lists the registers used to configure/query the miscellaneous features.

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
Fan Control Registers							
MFCR	0x9001	v	v		v	-	Table 1327, “MFCR - Management Fan Control Register Layout,” on page 1499
MFSC	0x9002	v	v		v	-	Table 1329, “MFSC - Management Fan Speed Control Register Layout,” on page 1500
MFSM	0x9003	v	v		v	-	Table 1331, “MFSM - Management Fan Speed Measurement Register Layout,” on page 1501
MFSL	0x9004	v	v		v	-	Table 1333, “MFSL - Management Fan Speed Limit Register Layout,” on page 1502
FORE	0x9007	v		v	v	-	Table 1335, “FORE - Fan Out of Range Event Register Layout,” on page 1503
MFNR	0x903B	v	-		v		Table 1337, “MFNR - Management Fan Part and Serial Number Register Layout,” on page 1504
Temperature, voltage and current monitoring							
MTCAP	0x9009	v	-		v	v	Table 1339, “MTCAP - Management Temperature Capabilities Register Layout,” on page 1505
MTECR	0x9109	v	-		v	v	Table 1341, “MTECR - Management Temperature Extended Capabilities Register Layout,” on page 1506
MTMP	0x900A	v	v		v	v	Table 1343, “MTMP - Management Temperature Register Layout,” on page 1507
MTWE	0x900B	v	-	v	v		Table 1345, “MTWE - Management Temperature Warning Event Register Layout,” on page 1510 [Internal]: for Switches, only SwitchX/-2
MTEWE	0x910B	v	-	v	v		Table 1347, “MTEWE - Management Temperature Extended Warning Event Register Layout,” on page 1511

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
MVCAP	0x902E	v	-		v		Table 1353, “MVCAP - Management Voltage and Current Capabilities Register Layout,” on page 1514
MVCR	0x900C	v	-		v		Table 1355, “MVCR - Management Voltage and Current Register Layout,” on page 1515 [Internal]: for Switches, only SwitchX/-2
MSPS	0x900D	v	-		v	-	Table 1357, “MSPS - Management System Power Supply Register Layout,” on page 1516
MCPD	0x900E	v	v		v		Table 1361, “MCPD - Management Control of Power Policer Register Layout,” on page 1518 [Internal]: for Switches, only Spectrum
MTBR	0x900F	v	-		v	-	Table 1349, “MTBR - Management Temperature Bulk Register Layout,” on page 1512
NVRAM (Flash) Access Registers							
MFPA	0x9010	v	v		v	v	Table 1363, “MFPA - Management Flash Parameters Access Register Layout,” on page 1520
MFBA	0x9011	v	v		v	v	Table 1365, “MFBA - Management Flash Block Access Register Layout,” on page 1522
MFBE	0x9012	-	v		v	v	Table 1367, “MFBE - Management Flash Block Erase Register Layout,” on page 1523
MFMC	0x9013	-	v		v	-	Table 1369, “MFMC - Management Flash Miscellaneous Configuration Register Layout,” on page 1524
MCIA	0x9014	v	v		v	v	Table 1371, “MCIA - Management Cable Info Access Register Layout,” on page 1526
MCAS - Management Cable Stamping	0x9015	v	v		v		Table 1373, “MCAS - Management Cable Stamping Register Layout,” on page 1530
MCION	0x9052	v	v		v	v	Table 1375, “MCION - Management Cable IO and Notifications Register Layout,” on page 1533
MQSP	0x9037	v			v	[Internal]	Table 1377, “MQSP - Module QSa Properties Register Layout,” on page 1534

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
MMIA	0x9016	v	v				Table 1379, “MMIA - Management Manufacturing Information Access Register Layout,” on page 1535
MMDIO - Management MDIO Interface	9017	v	v				Table 1381, “MMDIO - Management MDIO Register Layout,” on page 1536
MPGR	0x9019	v	v				Table 1383, “MPGR - Management Power Governor Register Layout,” on page 1537
MFM	0x901D	v	v		v		Table 1385, “MFM - Management Fabric Memory Register Layout,” on page 1538
MHSR	0x901E		v		v		Table 1387, “MHSR - Management Health State Register Layout,” on page 1539
MJTAG	0x901F	v	v		v	v	Table 1389, “MJTAG - Management JTAG Register Layout,” on page 1540
MSGI	0x9021	v			T B D		Table 1393, “MSGI - Management System General Information Register Layout,” on page 1542
MSCI	0x902A	v			v	v	Table 1395, “MSCI - Management System CPLD Information Register Layout,” on page 1543
MPCIR	0x905A	v	-		v		Table 1397, “MPCIR - Management and PHY Control for FW ISSU Flow Register Layout,” on page 1544
MVTS - Management Version Time-Stamp	0x902C	v	v		-	v	For NICs only [Internal]
MRTC	0x902D	v	v			v	Table 1423, “MRTC - Management Real Time Clock Register Layout,” on page 1561
MGIR	0x9020	v			v	v	Table 1399, “MGIR - Management General Information Register Layout,” on page 1545
MGPIR	0x9100	v	v		v		Table 1411, “MGPIR - Management General Peripheral Information Register Layout,” on page 1555
MDIR	0x911A	v	-		v		Table 1415, “MDIR - Management Device ID Register Layout,” on page 1557

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
MSSIR	0x9105	v	v		v		Table 1419, “MSSIR - Firebird Management System Status Information Register Layout,” on page 1558
MHMPR	0x905D	v	v		v	v	Table 1421, “MHMPR - Management Host Management of Ports Register Layout,” on page 1560
MCWD - Management CPU Watchdog Register	TBD	v	v				Table 1425, “MCWD - Management CPU Watchdog Register Layout,” on page 1562
MPFM	0x9022	v	v				Table 1427, “MPFM - Management PCIe Failure Mode Register Layout,” on page 1563
MRSR	0x9023	-	v		v	-	Table 1429, “MRSR - Management Reset and Shutdown Register Layout,” on page 1564
MPGO	0x9052	v	v		-	v	Table 1431, “MPGO Register Layout,” on page 1565
MNVDA	0x9024	v	v		v	v	Table 1442, “MNVDA Register Layout,” on page 1571
MNVDI	0x9025		v		v	v	Table 1461, “MNVDI Register Layout,” on page 1578
MNVQC	0x9030	v			v	v	Table 1475, “MNVQC Register Layout,” on page 1586
MNVCT	0x9032		v			v	Section 28.27.7, “MNVCT - Non-volatile commit,” on page 1585
MNVIA	0x9033		v		v	v	Table 1464, “MNVIA Register Layout,” on page 1579
MNVGC	0x9034	v			v	v	Table 1466, “MNVGC Register Layout,” on page 1580
MNVGN	0x9035	v				v	Table 1469, “MNVGN Register Layout,” on page 1581
MGNLE	0x9036	v				v	Table 1471, “MGNLE Register Layout,” on page 1583
Tracer_Registers	0x9040-0x904F	v	v		-	v	Table 1477, “Tracer Registers Summary,” on page 1587

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
MPPF	0x9049	v				v[internal]	Table 1506, “MPPF - Management PCIE Physical Functions Register Layout,” on page 1610
MLCR	0x902B	v	v		v	v	Table 1433, “MLCR - Management LED Control Register Layout,” on page 1566 [Internal]: for Switches, only Spectrum
MGCR	0x903A	v	-		v	v	Table 1435, “MGCR - Management GPIO Capabilities Register Layout,” on page 1567
MERR	0x903C	v	-		v	v	Table 1437, “MERR - Management Entities Remapping Register Layout,” on page 1569
MPEIN	0x9050	v	-		v	v	Table 1488, “MPEIN - Management PCIE Info Register Layout,” on page 1593 [Internal]: for Switches, only Spectrum
MPCNT	0x9051	v	-		v	v	Table 1490, “MPCNT - Management PCIE Counters Register Layout,” on page 1597 [Internal]: for Switches, only Spectrum
MTPPS	0x9053	v				v[internal]	Table 1508, “MTPPS - Management PulsePer Second Register Layout,” on page 1611
MTPPSE	0x9054	v	v			v[internal]	Table 1510, “MTPPSE - Management Pulse Per Second Event Register Layout,” on page 1616
MTUTC	0x9055						Table 1512, “MTUTC - Management UTC Register Layout,” on page 1617
MPEGC	0x9056	v	v			v	Table 1498, “MPEGC - Management PCIE General Configuration Register Layout,” on page 1604
MPiR	0x9059	v	v		v	v	Table 1500, “MPiR - Management PCIE Index Register Layout,” on page 1606
MPECS	0x905B	v	v		v	v	Table 1502, “MPECS - Management PCIE Configuration Space Register Layout,” on page 1607
MCQS	0x9060	v			v	v	Table 1514, “MCQS Register Layout,” on page 1619

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
MCQI	0x9061	v			v	v	Table 1516, “MCQI Register Layout,” on page 1622
MCC	0x9062	v	v		v	v	Table 1526, “MCC Register Layout,” on page 1629
MCDA	0x9063	v	v		v	v	Table 1528, “MCDA Register Layout,” on page 1633
MQIS	0x9064	v			v	v	Table 1530, “MQIS Register Layout,” on page 1634
MIRC	0x9162	v	v		v	v	Table 1532, “MIRC Register Layout,” on page 1635
MCDD	0x905C	v	v		v	v	Table 1534, “MCDD - Management Component DMA Based Data Access Register Layout,” on page 1636
MDFCR	0x9101	v	v		v		Table 1536, “MDFCR - Management Peripheral Devices Firmware Control Register Layout,” on page 1637
MDCR	0x9111	v	v		v		Table 1538, “MDCR - Management Devices Control Register Layout,” on page 1639
MDRCR	0x9102	v	v		v		Table 1542, “MDRCR - Management Peripheral Devices Reset Control Register Layout,” on page 1641
MDSR	0x9110	v	-		v		Table 1544, “MDSR - Management Debug Session Register Layout,” on page 1642
MRMR - Management Resource Monitoring Register	TBD	v					Table 1439, “MRMR - Management Resource Monitoring Register Layout,” on page 1570
MISU - Management Internal Stack Register	90FF	v	-	-	v		Table 1504, “MISU - Management Internal Stack Utilization Register Layout,” on page 1609
MCAM	0x907F	v	-	-		v[Internal]	Table 1546, “MCAM - Management Capabilities Mask Register Layout,” on page 1643
MFRL - Management Firmware Reset Level	0x9028	v	v			v	Table 1548, “MFRL - Management Firmware Reset Level Register Layout,” on page 1645

Table 1326 - Management and Peripherals Control Register Summary

Register Name	Register ID (hex)	Query	Write	Event	Switch	HCA	Reference
SWITCH ONLY	0x9080-0x909F						
SWITCH ONLY	0x90F0-0x90FE						

28.23 Fan Registers

28.23.1 MFCR - Management Fan Control Register

[Internal] I assume there is no need for enable bit and the information comes from the ini file.

This register controls the settings of the Fan Speed PWM mechanism:

Table 1327 - MFCR - Management Fan Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1328 - MFCR - Management Fan Control Register Fields

Offset	Bits	Name	Description	Access
00h	6:0	pwm_frequency	Controls the frequency of the PWM signal. Supported frequencies are: 0x0: 11Hz 0x1: 14.7Hz 0x2: 22.1Hz 0x3 - 0111111: Reserved 0x40: 1.4KHz 0x41: 5KHz 0x42: 20KHz 0x43: 22.5KHz 0x44: 25KHz 0x45 - 0x7F: Reserved	RW
04h	25:16	tacho_active	Indicates which of the tachometer is active (bit per tachometer)	RO
	4:0	pwm_active	Indicates which of the PWM control is active (bit per PWM)	RO

28.23.2 MFSC - Management Fan Speed Control Register

[Internal] I assume there is no need for enable bit and the information comes from the ini file.

This register controls the settings of the Fan Speed PWM mechanism:

Table 1329 - MFSC - Management Fan Speed Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1330 - MFSC - Management Fan Speed Control Register Fields

Offset	Bits	Name	Description	Access
00h	26:24	pwm	Fan pwm to control / monitor	Index
04h	7:0	pwm_duty_cycle	Controls the duty cycle of the PWM. Value range from 0..255 to represent duty cycle of 0%...100%	RW

28.23.3 MFSM - Management Fan Speed Measurement

This register controls the settings of the Tacho measurements and enables reading the Tachometer measurements.

Table 1331 - MFSM - Management Fan Speed Measurement Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				tacho																										n	00h	
																rpm											04h					

Table 1332 - MFSM - Management Fan Speed Measurement Register Fields

Offset	Bits	Name	Description	Access
00h	27:24	tacho	Fan tachometer index	Index
	1:0	n	[Internal] Defines the number of tacho pulses duration in which the SwitchX counts TBD mSec periods. FW should ignore this field and it will be removed in the future.	RW
04h	15:0	rpm	Fan speed (round per minute) calculated based on the time measurement between n fan pulses. Note that in order for the RPM to be correct, the n value should correspond to the number of tach pulses per rotation measured by the tachometer.	RO

28.23.4 MFSL - Management Fan Speed Limit

[TBD] Do we want to define it as speed limit or should we leave it in the same measure as the tachometer counter above?

The Fan Speed Limit register is used to configure the fan speed event / interrupt notification mechanism. Fan speed threshold are defined for both under-speed and over-speed.

Table 1333 - MFSL - Management Fan Speed Limit Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
					tacho																								ee		ie	00h
																tach_min												04h				
																tach_max												08h				

Table 1334 - MFSL - Management Fan Speed Limit Register Fields

Offset	Bits	Name	Description	Access
00h	27:24	tacho	Fan tachometer index	Index
	3:2	ee	Event Enable. 0 - Do not generate event 1 - Generate Event 2 - Generate single event For Fan Error Evet details refer to Section 28.23.5, "FORE - Fan Out of Range Event" .	RW
	0	ie	Interrupt Enable. If the ie bit is set and the Tachometer reading is out of range as defined by tach_min and tach_max, the FERR# will be asserted.	RW
04h	15:0	tach_min	Tachometer minimum value (minimum RPM).	RW
08h	15:0	tach_max	Tachometer maximum value (maximum RPM).	RW

[TODO] Need to define the interrupt configuration and Event notification packet format.

28.23.5 FORE - Fan Out of Range Event

The Fan Out of Range Event register reports the status of the controlled fans compared to the range defined by the MFSL register.

Table 1335 - FORE - Fan Out of Range Event Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
						fan_under_limit																												00h		
						fan_over_limit																														04h
																																	08h			

Table 1336 - FORE - Fan Out of Range Event Register Fields

Offset	Bits	Name	Description	Access
00h	25:16	fan_under_limit	Fan speed is below the low limit defined in MFSL register. Each bit relates to a single tachometer and indicates the specific tachometer reading is below the threshold.	RO
04h	25:16	fan_over_limit	Fan speed is above the high limit defined in MFSL register. Each bit relates to a single tachometer and indicates the specific tachometer reading is above the threshold.	RO

28.23.6 MFNR - Management Fan Part and Serial Number Register

The register allows to query the serial and the part numbers of the fans.

Table 1337 - MFNR - Management Fan Part and Serial Number Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																											fan_index				00h	
serial_number																															04h-18h	
part_number																															1Ch-2Ch	

Table 1338 - MFNR- Management Fan Part and Serial Number Register Fields

Offset	Bits	Name	Description	Access
00h	4:0	fan_index	Fan tachometer index.	Index
04h-18h	192	serial_number	Fan serial number (ASCII string) Value of '0': Some error (Fan not present, disconnected etc.)	RO
1Ch-2Ch	160	part_number	Fan part number (ASCII string) Value of '0': Some error (Fan not present, disconnected etc.)	RO

28.24 Temperature, Voltage and Power Registers

28.24.1 MTCAP - Management Temperature Capabilities

This register exposes the capabilities of the device and system temperature sensing.

Table 1339 - MTCAP - Management Temperature Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch

Table 1340 - MTCAP - Management Temperature Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h	6:0	sensor_count	Number of sensors supported by the device This includes the ASIC and the ambient sensors. QSFP module sensors are not included.	RO
04h	6:0	internal_sensor_count	Number of sensors supported by the device that are in the ASIC.	RO
08h-0Ch	64	sensor_map	Mapping of system sensors supported by the device. Only ASIC and ambient sensors are supported. Each bit represents a sensor. 0: Not connected or not supported 1: Supports temperature measurements	RO

28.24.2 MTECR - Management Temperature Extended Capabilities Register

This register exposes the capabilities of the device and system temperature sensing.

This register replaces MTCAP by exposing all possible temperature sensors that are on the system. **[Internal]: The maximum size of the number of sensors that this register is able to display is up to 2880 registers.**

Table 1341 - MTECR - Management Temperature Extended Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				last_sensor															sensor_count										00h			
																									internal_sensor_count					04h		
sensor_map[0]																															08h-5Ch	

Table 1342 - MTECR - Management Temperature Extended Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h	27:16	last_sensor	Last sensor index that is available in the system to read from.	RO
	11:0	sensor_count	Number of sensors supported by the device This includes the ASIC, ambient sensors, Gearboxes etc. QSFP module sensors are not included.	RO
04h	6:0	internal_sensor_count	Number of sensors supported by the device that are in the ASIC.	RO
08h-5Ch	31:0	sensor_map[0]	Mapping of system sensors supported by the device. Each bit represents a sensor. This field is size variable based on the <i>last_sensor</i> field and in granularity of 32bits. 0: Not connected or not supported 1: Supports temperature measurements	RO

28.24.3 MTMP - Management Temperature

This register controls the settings of the Temperature measurements and enables reading the Temperature measurements. Note that temperature is in 0.125 degrees Celsius.

Table 1343 - MTMP - Management Temperature Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
i																				sensor_index								00h				
																temperature											04h					
mte	mtr															max_temperature											08h					
tee																temperature_threshold_hi											0Ch					
																temperature_threshold_lo											10h					
																																04h
sensor_name_hi																												18h				
sensor_name_lo																												1Ch				

Table 1344 - MTMP - Management Temperature Register Fields

Offset	Bits	Name	Description	Access
00h	31	i	[Internal] Used to query the internal diodes of the switch ASIC; for i=1, sensor_index is the index of the ASIC diode. Those sensors are not available externally and should only be used for debug.	Index
00h	11:0	sensor_index	Sensors index to access. [internal] sensor index 63 is SYSMON 64-127 of sensor_index are mapped to the SFP+/QSFP modules sequentially (module 0 is mapped to sensor_index 64, module 1 to sensor_index 65 and so on). Gearbox sensors are from index 256.	Index
04h	31	mss	[Internal], TBD if implemented, May 2016 Module Sensor Supported 0: The module is not connected, or the module sensor is not supported 1: The module sensor is supported	RO
04h	15:0	temperature	Temperature reading from the sensor. Reading is in 0.125 Celsius degrees units. For negative values 2's complement is used (for example: -3.25 Celsius will read as 0xFFE6)	RO
08h	31	mte	Max Temperature Enable – enables measuring the max temperature on a sensor	RW
	30	mtr	Max Temperature Reset – clears the value of the max temperature register	OP
	15:0	max_temperature	The highest measured temperature from the sensor. When the bit mte is cleared, the field max_temperature is reserved.	RO

Table 1344 - MTMP - Management Temperature Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:30	tee	Temperature Event Enable 0 – do not generate event 1 – generate event 2 – generate single event Note: events may be delayed by up to 60 Seconds from “generate event” or “generate single event” Supported Only for switch devices. For HCA devices, SW should register Temperature Warning Event to an EQ.	RW
	15:0	temperature_ - threshold_hi	temperature_threshold_hi refers to the high threshold of Warning Event. If the sensor temperature measurement is above the threshold (and events are enabled), an event will be generated. threshold_hi and threshold_lo implements hysteresis mechanism of the threshold preventing toggling of the indication. Note that temperature_threshold_hi must be equal or lower than the system requirement. System requirement for module is the module warning temperature. System requirement for board/silicon sensors is according to product information parameters [Internal] INI. Note that the temperature threshold can be used to generate an event message or an interrupt using GPIO	RW
10h	15:0	temperature_ - threshold_lo	temperature_threshold_lo refers to the low threshold of Warning Event. The offset threshold_lo implements the lower threshold for the hysteresis mechanism of over temperature alert. Once alert is set, if the temperature goes below this threshold, the alert is cleared. Note that temperature_threshold_lo must be at least 5 degrees lower than temperature_threshold_hi	RW
18h	31:0	sen- sor_name_hi	Sensor Name 8 character long sensor name	RO
1Ch	31:0	sen- sor_name_lo	Sensor Name 8 character long sensor name	RO

28.24.4 MTWE - Management Temperature Warning Event

This register is used for over temperature warning.

Table 1345 - MTWE - Management Temperature Warning Event Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sensor_warning																															00h-0Ch	

Table 1346 - MTWE - Management Temperature Warning Event Register Fields

Offset	Bits	Name	Description	Access
00h-0Ch	128	sensor_warning	Bit vector indicating which of the sensor reading is above threshold. Address 00h bit31 is sensor_warning[127] Address 0Ch bit0 is sensor_warning[0]	RO

28.24.5 MTEWE - Management Temperature Extended Warning Event

This register is used for over temperature warning. This register replaces MTWE by exposing all possible temperature sensors that are on the system. **[Internal]: The maximum size of the number of sensors that this register is able to display is up to 2880 sensors..**

Table 1347 - MTEWE - Management Temperature Extended Warning Event Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
last_sensor																																04h-58h
sensor_warning[0]																																

Table 1348 - MTEWE - Management Temperature Extended Warning Event Register Fields

Offset	Bits	Name	Description	Access
00h	27:16	last_sensor	Last sensor index that is available in the system to read from.	RO
	11:0	sensor_count	Number of sensors supported by the device This includes the ASIC, ambient sensors, Gearboxes etc. QSFP module sensors are not included.	RO
04h-58h	31:0	sensor_warning[0]	Bit vector indicating which of the sensor reading is above threshold. This field is size dependent and based on <i>last_sensor</i> field and in granularity of 32bits..	RO

28.24.6 MTBR - Management Temperature Bulk Register

This register is used for bulk temperature reading.

Table 1349 - MTBR - Management Temperature Bulk Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch
																																10h
																																14h

Table 1350 - MTBR - Management Temperature Bulk Register Fields

Offset	Bits	Name	Description	Access
00h	11:0	base_sensor_index	Base sensor index Sensors are read starting from <i>base_sensor_index</i> Sensor 0 is the ASIC sensor, sensors 1-63 are ambient sensors and sensors 64-127 are mapped to the SFP+/QSFP modules sequentially (module 0 is mapped to sensor 64, module 1 to sensor 65 and so on). Gearbox sensors starts from 256 index.	Index
04h	7:0	num_rec	Number of records to read Range 1..255	OP
10h-14h	31:0	temperature_record[2] (See Table 1351, "MTBR - Temperature Record Layout," on page 1513)	Temperature record See Table 1351, "MTBR - Temperature Record Layout," on page 1513	RO

28.24.6.1 MTBR Temperature Record

Table 1351 - MTBR - Temperature Record Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h
max_temperature																temperature																

Table 1352 - MTBR - Temperature Record Fields

Offset	Bits	Name	Description	Access
00h	31:16	max_tempera- ture	The highest measured temperature from the sensor. When the bit relevant MTMP. <i>mte</i> is cleared, the field max_tem- perature is reserved. The value will be '0' for any of the '800X' temperature values and for '0' value.	RO
	15:0	temperature	Temperature reading from the sensor. Units of 0.125 Celsius degrees. If no ambient sensor is present then the value is '0'. Also for a passive copper case. 0x8000: NO_CONN- No cable is connected. 0x8001: NO_TEMP_SENS - In case a cable is connected but the temperature sensing is not supported. 0x8002: INDX_NA- No such cable index. Bigger than the number of modules. 0x8003: INCORRECT_SENS_INFO - Sensor information is incorrect. [Internal] We don't trust it. FDR cable for example.	RO

28.24.7 MVCAP - Management Voltage and Current Capabilities

The register exposes the capabilities of the device and system sensors that provide reading of voltage and current measurements.

Table 1353 - MVCAP - Management Voltage and Current Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sensor_map																																00h-04h
																																08h
																																0Ch

Table 1354 - MVCAP - Management Voltage and Current Capabilities Register Fields

Offset	Bits	Name	Description	Access
00h-04h	64	sensor_map	Mapping of system sensors supported by the device. Each bit represents a sensor. 0: Not connected or not supported 1: Supports voltage and current measurements	RO

28.24.8 MVCR - Management Voltage and Current Register

This register controls the settings of the system Voltage and Current measurements and enables reading the voltage and current measurements.

Table 1355 - MVCR - Management Voltage and Current Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h-14h

Table 1356 - MVCR - Management Voltage and Current Register Fields

Offset	Bits	Name	Description	Access
00h	6:0	sensor_index	Sensors index to access Mapping of system sensors to index is system specific, and detail can be found in the system specification. [Internal] The mapping is described in the Firmware INI file. 64-127 of sensor_index are mapped to the SFP+/QSFP modules sequentially (module 0 is mapped to sensor_index 64, module 1 to sensor_index 65 and so on).	Index
04h	15:0	voltage_sensor_value	Voltage reading from the sensor. Voltage units are 0.01V.	RO
08h	15:0	current_sensor_value	Current reading from the sensor.	RO
10h-14h	64	sensor_name	Sensor Name 8 character long sensor name	RO

28.24.9 MSPS - Management System Power Supply Register

This register enables monitoring the system power supply state.

Table 1357 - MSPS - Management System Power Supply Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
psu0																																00h-4Ch
psu1																																50h-9Ch

Table 1358 - MSPS - Management System Power Supply Register Fields

Offset	Bits	Name	Description	Access
00h-4Ch	640	psu0	System Power Supply 0 Status	RO
50h-9Ch	640	psu1	System Power Supply 1 Status	RO

This register enables monitoring the system power supply state.

Table 1359 - System Power Supply Status Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
present						alert																		dc_status				ac_status		00h		
																										temp				fan		04h
power_cap																power_consumption																08h
																														snp		0Ch
serial_number																																10h-24h
																																28h-2Ch
																																30h-40h

Table 1359 - System Power Supply Status Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																44h-4Ch

Table 1360 - System Power Supply Status Fields

Offset	Bits	Name	Description	Access
00h	31:28	present	System Power Supply present 0 – Information not available 1 – Present, not field replaceable 2 – Not present, not field replaceable 5 – Present, field replaceable 6 – Not present, field replaceable [Internal] Present is not supported in Dingo (PS is not an FRU)	RO
	25:24	alert	PSU Alert 0 - Information not available 1 - No Alert 2 - Alert Set	RO
	5:4	dc_status	DC Power OK 0 – Information not available 1 – DC Power OK 2 – DC Power Error [Internal] Supported in all systems	RO
	1:0	ac_status	AC Power Exist 0 – Information not available 1 – AC power exist 2 – AC power do not exist [Internal] Supported in Dingo, not supported in Stingray / Piranha	RO
04h	5:4	temp	PSU Thermal Status 0 – Information not available 1 – Temperature OK 2 – Over Temperature Error [Internal] Will only be supported in Piranha-X / Coyote-X and EDR systems	RO
	1:0	fan	PSU Fan Status 0 – Information not available 1 – No PSU fan 2 – PSU fan OK 3 – PSU fan error [Internal] No PS fan in Dingo. Supported in other systems	RO

Table 1360 - System Power Supply Status Fields

Offset	Bits	Name	Description	Access
08h	31:30	power_cap	Capability bit 0 - Information not available [Internal]: feature is not supported 1 - power_consumption field is not supported for that PSU. 2- power_consumption field is supported. 3- Error in read	RO
	15:0	power_consumption	Input power consumption. In Watts.	RO
0Ch	0	snp	Serial Number Present. Indicates the serial_number field is valid. If cleared, serial_number information is not available	RO
10h-24h	192	serial_number	System serial number (ASCII string) [Internal] Does not exist in Dingo (not FRU)	RO
30h - 40h	160	part_number	System part number (ASCII string)	RO

28.24.10MCP - Management Control of Power Policer

The MCP register is used to control the power policer.

Table 1361 - MCP - Management Control of Power Policer Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ent	enoc																															00h
								warning_temperature																hysteresis_temperature								04h
wst	wsoc																															08h
								red_state_rate																yellow_state_rate								0Ch

Table 1362 - MCPP - Management Control of Power Policer Register Fields

Offset	Bits	Name	Description	Access
00h	31	ent	Enable Temperature Events Enable power policer according to high temperature 0 - disable (default) 1 - enable	RW
	30	enoc	Enable Over Current Events Enable power policer according to indication of Over Current which is provided by device interface (e.g. OC_WARN pin in Spectrum) 0 - disable 1 - enable (default)	RW
04h	23:16	warning_tem- perature	Above this temperature the Temperature Event is set. Units are Celsius Default 105 Reserved when <i>ent</i> = 0	RW
	7:0	hysteresis_tem- perature	Below this temperature the Temperature Event is cleared. Units are Celsius Default 95 <i>hysteresis_temperature</i> must be lower or equal to <i>warning_tem- perature</i> Reserved when <i>ent</i> = 0	RW
08h	31	wst	Warning State due to Temperature Event: 0 - Yellow_State (default) 1 - Red_State	RW
	30	wsoc	Warning State due to Over Current Event: 0 - Yellow_State 1 - Red_State (default)	RW
0Ch	23:16	red_state_rate	Port rate at Red_State. Units are percentage from Green_State port rate <i>red_state_rate</i> must be lower or equal to <i>yellow_state_rate</i> Default 0	RW
	7:0	yel- low_state_rate	Port rate at Yellow_State. Units are percentage from Green_State port rate Default 50	RW

28.25 Flash Registers

This section include the registers used to read, write and erase flash data.

28.25.1 MFPA - Management Flash Parameters Access

The MFPA register is used to query the attached flash parameters.

Table 1363 - MFPA - Management Flash Parameters Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
add cap 32b																								p			fs						00h	
boot_address																															04h			
																															08h-0Ch			
wip		bulk 32kb erase_en		bulk 64kb erase_en																											flash_num	10h		
								jedec_id																							14h			
block_size								block_alignment												sector_size							18h							
capability_mask																															1Ch			

Table 1364 - MFPA - Management Flash Parameters Access Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	add_cap_32b	Capability bit - If set to '1', <i>boot_address</i> field is 32 bit length.	RO
00h	8	p	[Internal] Parallel. Not supported. Must always set to '0'.	RO
00h	5:4	fs	Flash select - selects the flash device. Only zero is supported for NICs with a single flash device. Range between 0 .. <i>flash_num</i> -1 [Internal] older firmware versions also didn't fail fs==1 in order not to break old tools.	Index
04h	31:0	boot_address	The flash address from which the firmware will boot in the next reset (warm start).	RW

Table 1364 - MFPA - Management Flash Parameters Access Field Descriptions

Offset	Bits	Name	Description	Access
10h	31	wip	Work In Progress. Indicates that the flash is currently busy.	RO
	30	bulk_32kb_erase_en	If set, MFBE register supports 32KB bulk erase operation.	RO
	29	bulk_64kb_erase_en	If set, MFBE register supports 64KB bulk erase operation.	RO
	17	sector_wrp_en	If set, MFMC register supports setting write protect with sector blocks. Sector block size is available in MFMC. <i>sector_protect_size</i>	RO
	16	sub_sector_wrp_en	If set, MFMC register supports setting write protect with sub-sector blocks Sub-sector block size is available in MFMC. <i>sub_sector_protect_size</i>	RO
	3:0	flash_num	The number of flash devices 1: there is 1 flash device, etc.	RO
14h	23:0	jedec_id	Return the flash JEDEC ID value returned by the standard Read JEDEC ID command that is available in most flash devices.	RO
18h	31:24	block_size	The block size in bytes of the flash device, max read size for MFBA (typically 128 bytes).	RO
	23:16	block_alignment	Log 2 of the requested write alignment in bytes. Write access to the flash must not cross this alignment. Read and write access must be aligned to this value.	RO
	9:0	sector_size	The minimal unit that can be erased with and MFBE command (in bytes).	RO
1Ch	31:0	capability_mask	Bitmask indicates which capabilities are supported. Bit 15..0: reserved [internal] FW bug in old FW versions returned those bits set. [internal] Bit 0: parallel flash support (never implemented) Bit 16: Set write protect supported (register MFMC supports setting write protection) Bit 17: quad enable read write is supported (register MFMC supports setting quad enable) Bit 18: set dummy cycle supported (register MFMC supports setting dummy cycles)	RO

28.25.2 MFBA - Management Flash Block Access

The MFBA register is used to read and write data to the flash

Table 1365 - MFBA - Management Flash Block Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
add cap 32b																							p			fs						00h		
																							size											04h
	address																															08h		
	data																															0Ch-108h		

Table 1366 - MFBA - Management Flash Block Access Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	add_cap_32b	Capability bit - If set to '1', <i>address</i> field is 32 bit length.	RO
	8	p	[Internal] Parallel. Not supported. Must always set to '0'.	WO
	5:4	fs	Flash select - selects the flash device. Only zero is supported for NICs with a single flash device Range between 0 .. <i>MFPA.flash_num</i> -1	Index
04h	8:0	size	The size of the data to be written or read in bytes Should comply with <i>block_size</i> and <i>block_alignment</i> fields in MFPA.	WO
08h	31:0	address	The start address of this read or write access in bytes. Should comply with <i>block_size</i> and <i>block_alignment</i> fields in MFPA.	Index
0Ch-108h	2048	data	The data to be written or read data.	RW

28.25.3 MFBE - Management Flash Block Erase

The MFBE register is used to erase areas in the Flash.

Table 1367 - MFBE - Management Flash Block Erase Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
			bulk 64kb erase																						p			fs					00h
																															04h		
address																															08h		

Table 1368 - MFBE - Management Flash Block Erase Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	add_cap_32b	Capability bit - If set to '1', <i>address</i> field is 32 bit length.	RO
00h	30	bulk_32kb_erase	Erase a 32KB flash area in one bulk operation.	WO
	29	bulk_64kb_erase	Erase a 64KB flash area in one bulk operation.	WO
	8	p	[Internal] Parallel. Not supported. Must always set to '0'.	
	5:4	fs	Flash select - selects the flash device. Only zero is supported for NICs with a single flash device. Range between 0 .. <i>MFWA.flash_num</i> -1	Index
08h	31:0	address	The start address (in bytes) of this erase operation. Must be aligned with the selected erase size (sector, 32KB or 64KB).	Index

28.25.4 MFMC - Management Flash Miscellaneous Configuration Register

[Internal], July 2016, Switch FW supports:

1. Windbond (W25Q32BV/W25Q128BV)

2. Spansion (S25FL132K0XMFI013)

Table 1369 - MFMC - Management Flash Miscellaneous Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																										fs						00h
																																04h
																																08h
																																0Ch
																																10h
																																14h
																																18h

Table 1370 - MFMC - Management Flash Miscellaneous Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	5:4	fs	Flash select - selects the flash device. Only zero is supported for NICs with a single flash device. Range between 0 .. <i>MFPA.flash_num</i> -1	Index

Table 1370 - MFMC - Management Flash Miscellaneous Configuration Register Fields

Offset	Bits	Name	Description	Access
04h	31	wrp_en	Write protect enable Set write protect of flash device	RW
	17:16	block_size	Block size 0: write protect sub-sector blocks 1: write protect sector blocks Reserved when <i>wrp_en</i> = 0 Note that not all block sizes are supported on all flash device, need to check MFPA capabilities	RW
	7:0	wrp_block_count	Power of 2 of the write protect block count 0: 1 block 1: 2 blocks 2: 4 blocks 3: 8 blocks etc. Range 0..5 Note that per flash device there may be invalid configurations Reserved when <i>wrp_en</i> = 0	RW
08h	13:8	sector_protect_size	Power of 2 for sector size in 4Kbytes. 0: 4Kbyte. 1: 8 Kbyte 2: 16Kbyte. Etc.	RO
	5:0	sub_sector_protect_size	Power of 2 for sub_sector size in 4Kbytes. 0: 4Kbyte. 1: 8 Kbyte 2: 16Kbyte. Etc.	RO
10h	24	quad_en	Quad enable IO bit in the device status register	RW
18h	3:0	dummy_clock_cycles	The number of dummy clock cycles subsequent to all FAST READ commands. Reserved if not supported by the device	RW

28.26 Peripheral Registers

28.26.1 MCIA - Management Cable Info Access

MCIA register is used to access the SFP+ and QSFP connector's EPROM.

Table 1371 - MCIA - Management Cable Info Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
1								module																		status								00h
i2c_device_address								page_number								device_address																04h		
passwd_v	passwd_clr	passwd_cap						bank_number								size																08h		
			password																															
dword[0]																																10h		
dword[1]																																14h		
dword[2]																																18h		
dword[3]																																1Ch		
dword[4]																																20h		
dword[5]																																24h		
dword[6]																																28h		
dword[7]																																2Ch		
dword[8]																																30h		
dword[9]																																34h		
dword[10]																																38h		
dword[11]																																3Ch		
dword[12]																																40h		
dword[13]																																44h		
dword[14]																																48h		
dword[15]																																4Ch		
dword[16]																																50h		
dword[17]																																54h		
dword[18]																																58h		

Table 1371 - MCIA - Management Cable Info Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dword[19]																																5Ch
dword[20]																																60h
dword[21]																																64h
dword[22]																																68h
dword[23]																																6Ch
dword[24]																																70h
dword[25]																																74h
dword[26]																																78h
dword[27]																																7Ch
dword[28]																																80h
dword[29]																																84h
dword[30]																																88h
dword[31]																																8Ch

Table 1372 - MCIA - Management Cable Info Access Register Fields

Offset	Bits	Name	Description	Access
00h	31	1	Lock bit. Setting this bit will lock the access to the specific cable. Used for updating a full page in a cable EPROM. Any access other then subsequence writes will fail while the port is locked.	RW
	23:16	module	Module number	Index
	7:0	status	Module Status 0h - GOOD 1h - NO_EEPROM_MODULE. No response from module's EPROM. 2h - MODULE_NOT_SUPPORTED. Module type not supported by the device. [Internal] we currently supports all module but who knows. 3h - MODULE_NOT_CONNECTED. No module present indication. [Internal] The below codes are available in current FW. I think is should be removed. 4h - MODULE_TYPE_INVALID - if the module is not qsfp or sfp - meaning it doesn't has eeprom by definition (no need to check the bus as NO_EEPROM_MODULE). Module types comes from ini. 5h - MODULE_NOT_ACCESSIBLE - this is future implemented but not tested flow. If a module causing the I²C to hang (while we access its module) the firmware reset the I²C and remembers not to access this module until it is unplugged. we should fail access commands for this module. 9h - I²C_ERROR. Error occurred while trying to access the module's EPROM using I²C. 10h - MODULE_DISABLED - module is disabled [Internal] Triggers are: 1. Disable Command. 2. Thermal shut-down 3. Over current. [Internal] 11h - Page Access Failed - page not supported or trying to access a protected page with bad password.	RO
04h	31:24	i2c_device_address	I ² C device address	RW
	23:16	page_number	Page number	RW
	15:0	device_address	Device address	RW

Table 1372 - MCIA - Management Cable Info Access Register Fields

Offset	Bits	Name	Description	Access
08h	31	passwd_v	If set to 1 the Password field is valid and is written to the module password field (address 123-126) before accessing the page. This field is reserved if passwd_cap is not set to 1. Note: for SFP modules the password should be set only with i2c_device_address = A2h.	OP
	30	passwd_clr	If set to 1 will force the password to be cleared from the module password field at the end of the module access. This field is reserved if passwd_cap is not set to 1. Note: for SFP modules the password should be set only with i2c_device_address = A2h	OP
	29	passwd_cap	Password capability	RO
08h	23:16	bank_number	Bank number	
	15:0	size	Number of bytes to read/write (up to 128 bytes)	RW
0Ch	31:0	password	The password that is written to the module password field. This field is reserved if <i>passwd_cap</i> is not set to 1.	OP
10h-8Ch	31:0	dword[32]	Bytes to read/write	RW

28.26.2 MCAS - Management Cable Stamping

[Internal] This register is supported by FW for Mellanox usage

MCAS Register is used to expose cable stamping status and enables bypassing the stamping mechanism. This register is accessible via the ACCESS_REG command.

Table 1373 - MCAS - Management Cable Stamping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
								module																									00h
stamp- ing_select								stamp- ing_en						fdr_ cable_ _sta- tus	edr_ cable_ _sta- tus	lr_ca- ble_ _sta- tus	eth_1 00g_ cable_ _sta- tus	lr4_1 00g_ cable_ _sta- tus	sfp_2 5g_ cable_ _sta- tus	hdr_ cable_ _sta- tus	eth_4 00g_ cable_ _sta- tus									04h			
																																08h-0Ch	

Table 1374 - MCAS - Management Cable Stamping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number	Index

Table 1374 - MCAS - Management Cable Stamping Register Fields

Offset	Bits	Name	Description	Access
04h	31:28	stamping_select	Stamping Select: 0 - FDR/FDR10 cable stamping 1 - LR4 40G Long range stamping 2 - EDR cable stamping 3 - Ethernet QSFP 200GE/100GE/50GE/25GE stamping 4 - LR4 100G (IB/ETH) Long range stamping 5 - Ethernet SFP 25GE stamping 6 - HDR cable stamping 7 - Ethernet QSFP-DD 400GE/200GE/100GE/50GE/25GE stamping	Index
	23	stamping_en	Enable / Disable cable stamping	RW
	17:16	fdr_cable_status	FDR Cable Stamping status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	15:14	edr_cable_status	EDR Cable Stamping status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	13:12	lr_cable_status	LR4 40G Long range Cable Status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	11:10	eth_100g_cable_status	Ethernet QSFP 200GE/100GE/50GE/25GE status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	9:8	lr4_100g_cable_status	LR4 100G (IB/ETH) Long Range status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	7:6	sfp_25g_cable_status	Ethernet SFP 25GE stamping: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO
	5:4	hdr_cable_status	HDR cable stamping status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO

Table 1374 - MCAS - Management Cable Stamping Register Fields

Offset	Bits	Name	Description	Access
	3:2	eth_400g_cable_status	Ethernet QSFP-DD 400GE/200GE/100GE/50GE/25GE stamping status: 0 - No cable plugged 1 - Cable plugged not supported 2 - Cable plugged supported	RO

28.26.3 MCION - Management Cable IO and Notifications Register

MCION register is used to query SFP and QSFP connector's I/O pins and other notifications.

Table 1375 - MCION - Management Cable IO and Notifications Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h
																																14h

Table 1376 - MCION - Management Cable IO and Notifications Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
04h	15:0	module_status_bits	Module IO status, as defined by SFF: Bit 0 - Present Bit 1 - RX_LOS Bit 2 - TX_Fault [Internal] Bit 3 - RX CDR Loss of Lock [Internal] Bit 4 - TX CDR Loss of Lock [Internal] Bit 5 - DataNotReady [Internal] Bit 6- LPMode [Internal] Bit 7- Interrupt (IntL) [Internal] Bit 8 - Low Power Mode Status	RO
0Ch	3:0	module_inputs	Module IO inputs: Bit 0- Reset Bit 1- TX_Disable Bit 2- RS0 Bit 3- RS1	RW
10h	3:0	module_inputs_mask	On write commands, indicates which of the bits from <i>module_inputs</i> field are updated. Reserved on read operations.	WO

28.26.4 [Internal] MQSP - Module QSa Properties Register

MQSP register is used to query QSA parameters.

Table 1377 - MQSP - Module QSa Properties Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
								module																		status								00h
																																04h		
																																08h		
qsa_device_id															qsa_nominal_bitrate																0Ch			
																qsa_link_modu_support																10h		
																																14h		
																attenuation_7g								attenuation_12g								18h		
																																1Ch		

Table 1378 - MQSP - Module QSa Properties Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	module	Module number.	Index
	7:0	status	Status of QSA: 0 - Valid QSA data 1 - No QSA connected	RO
0Ch	31:24	qsa_device_id	type of QSA connected 0 - QSA10 1 - QSA28	RO
	15:0	qsa_nominal_bitrate	QSA nominal bit rate in 100Mb/s units	RO
10h	15:0	qsa_link_modu_support	QSA supported speeds/protocols Bit 0- 10Gbd NRZ / SDR-QDR Bit 1- 14Gbd NRZ / FDR Bit 2- 25Gbd NRZ / EDR Bit 3- 25Gbd PAM4 / HDR	RW
18h	15:8	attenuation_7g	QSA attenuation @ 7GHz	RW
	7:0	attenuation_12g	QSA attenuation @ 12.89/13.2GHz	RW

28.26.5 MMIA - Management Manufacturing Information Access

MMIA register is used access the system manufacturing Information.

Table 1379 - MMIA - Management Manufacturing Information Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																operation	00h
																																	04h
Data																																	08h

Table 1380 - MMIA - Management Manufacturing Information Access Register Fields

Offset	Bits	Name	Description	Access
00h	2:0	operation	0 - NOP 1 - Erase 2 - Read 3 - Write	OPWO
08h	31:0	Data	<u>TBD</u> <u>This field should include the manufacturing data stored in the VPD.</u> <u>Should include serial number and more.</u>	RW

28.26.6 MMDIO - Management MDIO Interface

MMDIO Register is used access PHY devices through the MDIO interface.

Table 1381 - MMDIO - Management MDIO Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h

Table 1382 - MMDIO - Management MDIO Register Fields

Offset	Bits	Name	Description	Access
00h	19:16	mdio_index	Selection of the MDIO interface	Index
	9:8	clause	MDIO Definition: 0 - Clause 22 1 - Clause 45	WO
	2:0	operation	0 - NOP 1 - Address (reserved for Clause 22) 2 - Read 3 - Write 4 - Post Read Increment Address (reserved for Clause 22) 6 - Address + Read - Generates Address cycle and then Read cycle in Clause 45 (reserved for Clause 22) 7 - Address + Write - Generates Address cycle and then Write cycle in Clause 45 (reserved for Clause 22)	WO
04h	5:0	reg_adr	Reg Address (Clause 22) / Dev Type (Clause 45)	RW
08h	15:0	data	Data (Clause 22) / Address/Data (Clause 45)	RW
	31:16	address	Address (Clause 45) This field is only valid for Address + Read and Address + Write operations, providing the address. For other Clause 45 operations the data field provides the address when appropriate.	RW

28.26.7 MPGR - Management Power Governor Register

MPGR register is used to control and monitor SwitchX® Power Governor.

Table 1383 - MPGR - Management Power Governor Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												fsps_admin			00h	
																												fsps_status			04h	
																																08h-0Ch
																												vsps_admin			10h	
																												vsps_status			14h	
																																18h-1Ch

Table 1384 - MPGR - Management Power Governor Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	fsps_admin	Frequency Scaling Power Save Admin state: 0 - Disable Frequency Scaling - Force device to highest frequency supported 1 - Force Frequency Scaling - Force device to highest frequency supported ^a 2 - Auto - Automatic Frequency Scaling (based on port's speed).	RW
04h	3:0	fsps_status	Actual Frequency 0 - High Frequency 1 - Low Frequency	RO
10h	3:0	vsps_admin	Voltage Scaling Power Save Admin state: 0 - Disable Voltage Scaling - Force device to high voltage 1 - Force Voltage Scaling - Force device to low voltage 2 - Auto - Automatic Voltage Scaling (based on device active frequency).	RW
14h	3:0	vsps_status	Actual Voltage 0 - High Voltage 1 - Low Voltage	RO

- a. Forcing the device to low frequency while one or more of the links are active at FDB speed will fail the command. Forcing the device to low frequency while none of the ports are at FDR will succeed and the ports will NOT be able to work at FDR speed.

28.26.8 MFM - Management Fabric Memory Register

MFM register is used to query and update the Fabric Memory content.

Table 1385 - MFM - Management Fabric Memory Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																													index			00h
																																04h
memory_high																																08h
memory_low																																0Ch
memory_mask_high																																10h
memory_mask_low																																14h

Table 1386 - MFM - Management Fabric Memory Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	index	Index to the fabric memory table.	Index
08h	31:0	memory_high	Holds software-written data. Default value: 0x0 Device reset will reset to the default value.	RW
0Ch	31:0	memory_low		
10h	31:0	memory_mask_high	On write commands, indicates which of the bits from <i>memory</i> field are updated. Reserved on read operations.	WO
14h	31:0	memory_mask_low		

28.26.9 MHSR - Management Health State Register

MHSR register allows software to update its health state.

Table 1387 - MHSR - Management Health State Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1388 - MHSR - Management Health State Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	health	Software health state: 0 - Failure 1 - Reserved 2 - Normal Operation 3 - Boot / Init (default state) Else - Reserved	WO

28.26.10MJTAG - Management JTAG Register

MJTAG register allows software to access the JTAG master interface.

Table 1389 - MJTAG - Management JTAG Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
cmd				sequence_number																				size								00h
jtag_transaction_set[0]								jtag_transaction_set[1]								jtag_transaction_set[2]								jtag_transaction_set[3]								04h
...																																...
jtag_transaction_set[40]								jtag_transaction_set[41]								jtag_transaction_set[42]								jtag_transaction_set[43]								2Ch

Table 1390 - MJTAG - Management JTAG Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	cmd	Command 0 – JTAG interface disable – JTAG interface is disabled 1 – JTAG interface enable – enabling JTAG interface 2 – JTAG interface access	RW
	27:24	sequence_number	Command sequence number – to prevent re-execution of the same command due to command retransmission, this command includes a sequence number. The enable and disable commands MUST be issues with sequence_number = 0 while any JTAG access command must increase the sequence_number by 1 (the first JTAG access command should start at sequence_number = 1).	WO
	7:0	size	Size of operation – relevant for JTAG interface access commands only, indicating the number of transaction bytes to execute. Maximum size supported is 40. Note: when size=0, no operation is performed (clock is not toggled). The tdo is sampled and provided in the response at jtag_transaction_set_0 .	WO
04h-2Ch	31:24	jtag_transaction_set[44]	See Table 1391, “JTAG Transaction Set Byte Layout,” on page 1540	

The tables below illustrates the layout and fields of the JTAG transaction set.

Table 1391 - JTAG Transaction Set Byte Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																											tdo		tdi	tms	00h	

Table 1392 - JTAG Transaction Set Byte Fields

Offset	Bits	Name	Description	Access
00h	3	tdo	JTAG tdo	RO
	1	tdi	JATG tdi	WO
	0	tms	JTAG tms	WO

28.26.11MSGI - Management System General Information Register

MSGI register allows software to query the system manufacturing information.

Table 1393 - MSGI - Management System General Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
serial_number																															00h-14h	
																																18h-1Ch
part_number																															20h-30h	
																																34h
revision																															38h	
manufacturing_date																															3Ch	
product_name																															40h-7Ch	

Table 1394 - MSGI - Management System General Information Register Fields

Off-set	Bits	Name	Description	Access
00h-14h	192	serial_number	System serial number (ASCII string)	RO
20h-30h	160	part_number	System part number (ASCII string)	RO
38h	31:0	revision	Revision (ASCII string)	RO
3Ch	31:0	manufacturing_date	Manufacturing Date	RO
40h-7Ch	512	product_name	Product Name (ASCII string)	RO

28.26.12MSCI - Management System CPLD Information Register

MSCI register allows querying for the System CPLD(s) information.

Table 1395 - MSCI - Management System CPLD Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
map_cap																									index			00h				
	version																											04h				
	cpld0_type				cpld1_type				cpld2_type				cpld3_type				cpld4_type				cpld5_type				cpld6_type				cpld7_type		08h	
																												08h				

Table 1396 - MSCI - Management System CPLD Information Register Fields

Offset	Bits	Name	Description	Access
00h	31	map_cap	[Internal] : If set to '1', cpld#_type fields are supported.	RO
00h	3:0	index	CPLD index	Index
04h	31:0	version	CPLD version	RO
08h	31:0	cpld#_type	[Internal] : A mapping of the CPLDs in the system which are read by FW. Up to 8 CPLDs are supported while 4 bits represent the CPLD type for each of those CPLDs (up to 15 types are supported). 0: N / A 1: Management CPLD 2: Switch board 3: Port type CPLD 4: Switch board with port 5: ComEx	RO

28.26.13 MPCIR - Management and PHY Control for FW ISSU Flow Register

The register allows to control the various management and PHY flows and services during FW ISSU flow. In addition, it allows to query the state of those services.

Table 1397 - MPCIR - Management and PHY Control for FW ISSU Flow Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
all																														00h		
																								ports	04h							
																																08h
																														ports_stat	0Ch	

Table 1398 - MPCIR- Management and PHY Control for FW ISSU Flow Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	all	If set to '01', activates the flow of preparation for FW ISSU, on all services. The values in op-codes for "per-service" are ignored. If set to '10', returns to operational state on all services. The values in op-codes for "per-service" are ignored. 11 - Reserved	OP
04h	1:0	ports	For each of the services, the following operations are available: 0: N/A (no action) 1: start preparation flow for FW ISSU 2: return to operational service (end of FW ISSU flow) 3: get status When set to '3', the current status will appear in corresponding <i>_stat</i> fields.	OP
0Ch	1:0	ports_stat	Status for each of the services. 0: not in FW ISSU flow state (FW ISSU flow is not initiated) 1: done with preparations for FW ISSU flow 2: Preparation for FW ISSU flow started but FW still not done service handling [Internal]: busy with some other critical flow)[Internal]: (busy with other critical flow, for example).[Internal]: (busy with other critical flow, for example).. For not implemented services on a particular system, the status should be '1'. [Internal]: (busy with other critical flow, for example).	RO

28.26.14MGIR - Management General Information Register

MGIR register allows software to query the hardware and firmware general information.

Note that for NIC devices, this register is supported according to MCAM register.

Table 1399 - MGIR - Management General Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hw_info(see Table 1401 , “Hardware Info Layout,” on page 1546)																																00h-1Ch
fw_info(see Table 1403 , “Firmware Info Layout,” on page 1548)																																20h-5Ch
sw_info(see Table 1405 , “Software Info Layout,” on page 1550)																																60h-7Ch
dev_info(see Table 1409 , “Development Info Layout,” on page 1554)																																80h-98h
reserved																																9Ch

Table 1400 - MGIR - Management General Information Register Fields

Offset	Bits	Name	Description	Access
00h-1Ch	256	hw_info(see Table 1401 , “Hardware Info Layout,” on page 1546)	Hardware Information, see Table 1401 , “Hardware Info Layout,” on page 1546	
20h-5Ch	512	fw_info(see Table 1403 , “Firmware Info Layout,” on page 1548)	Firmware Information, see Table 1403 , “Firmware Info Layout,” on page 1548	
60h-7Ch	256	sw_info(see Table 1405 , “Software Info Layout,” on page 1550)	Software Information, see Table 1405 , “Software Info Layout,” on page 1550 This field indicates the oldest software version compatible with the current firmware	

Table 1400 - MGIR - Management General Information Register Fields

Offset	Bits	Name	Description	Access
80h-98h	224	dev_info(see Table 1409, “Development Info Layout,” on page 1554)	[Internal]: Development Information, see Table 1409, “Development Info Layout,” on page 1554	

The details of each section of the GeneralInfo MAD are described in the tables below.

Table 1401 - Hardware Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
device_hw_revision																device_id																00h				
									num_ports																							pvs				04h
																hw_dev_id																08h				
																																0Ch				
																manufacturing_base_mac[47:32]																10h				
manufacturing_base_mac[31:0]																																14h				
																																18h				
uptime																																1Ch				

Table 1402 - Hardware Info Format

Offset	Bits	Name	Description	Access
00h	31:16	device_hw_revision	InfiniScale IV Devices: 0x00A0: Device step A0, FCC package 0x00A1: Device step A1, FCC package 0x01A1: Device step A1, FDC package SwitchX Devices: 0x00A0: Device step A0, FCC package 0x00A1: Device step A1, FCC package 0x00A2: Device step A2, FCC package Switch-IB Devices: 0x00A0: Switch-IB device step A0, FCC package 0x0000 is a valid value for some devices not included in PRM.	RO
	15:0	device_id	PCI device ID.	RO
04h	23:16	num_ports	Number of physical port the device supports.	RO
	4:0	pvs	Process Voltage Scaling Maximum supported V_CODE voltage (in 50mV increments) for the device. [Internal] Minimum V_CORE voltage required for FDR / 40GE operation. [Internal] For SwitchX and SwitchX-2, V_CORE operation is legal in the range of [(0.9V - 10%)..(dvfs + 10%)]	RO
08h	15:0	hw_dev_id	The PCI device-ID of the NIC/HCA in recovery (Live-fish) mode.	RO
10h	15:0	manufacturing_base_mac[47:32]	MSB of the 'base' MAC address of the NIC that was allocate during manufacturing. The NIC derives the MAC addresses for the different PCI PFs from this MAC address. This parameter can be used as a canonical unique identifier of the NIC. manufacturing_base_mac of value 0 means field is not supported.	RO
14h	31:0	manufacturing_base_mac[31:0]	LSB of the 'base' MAC address of the NIC that was allocate during manufacturing. The NIC derives the MAC addresses for the different PCI PFs from this MAC address. This parameter can be used as a canonical unique identifier of the NIC. manufacturing_base_mac of value 0 means field is not supported.	RO
1Ch	31:0	uptime	Time (in secs.) since last reset0	RO

Table 1403 - Firmware Info Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset	
			string_tlv	dev	debug	signed_fw	secured	major							minor							sub_minor										00h
build_id																														04h		
month							day							year																08h		
																	hour													0Ch		
psid[0]							psid[1]							psid[2]							psid[3]								10h			
...																														...		
psid[12]							psid[13]							psid[14]							psid[15]								1Ch			
ini_file_version																														20h		
extended_major																														24h		
extended_minor																														28h		
extended_sub_minor																														2Ch		
																isfu_major														30h		
																											sec_boot	life_cycle	34h			
																															38h-3Ch	

Table 1404 - Firmware Info Field Descriptions

Offset	Bits	Name	Description	Access
00h	28	string_tlv	When set, string-TLV is supported.	RO
	27	dev	When set, the device is running a development firmware version.	RO
	26	debug	When set, the device is running a debug firmware. 'debug' binary can only be installed on specific devices (identified by their 'Factory base MAC'), which currently run a specific firmware version. These restrictions are expressed by a signed 'debug' token that must be loaded to the device before installing the debug binary.	RO
	25	signed_fw	When set the device is running a signed FW binaries.	RO
	24	secured	When set, the device is running firmware with secure-firmware updates capabilities. [Internal] If 'secure' then the FW must be 'signed'.	RO
	23:16	major	Major firmware version number. Deprecated and returns '0'. Refer to extended_major.	RO
	15:8	minor	Minor firmware version number. Deprecated and returns '0'. Refer to extended_minor.	RO
	7:0	sub_minor	Sub-minor firmware version number. Deprecated and returns '0'. Refer to extended_sub_minor.	RO
04h	31:0	build_id	Firmware Build ID. Optional. [Internal] '0' for devices which do not support it.	RO
08h	31:24	month	Firmware installation date. For example: 3 May 2004 will be coded as Month= 0x05, Day= 0x03, and Year= 0x04	RO
	23:16	day		RO
	15:0	year		RO
0Ch	15:0	hour	Firmware installation hour. For example 17:43 will be coded as 0x1743	RO
10h-1Ch	31:24	psid[16]	To be defined later.	RO
20h	31:0	ini_file_version	User-configured version number of the current INI file.	RO
24h	31:0	extended_major	FW version's Major field in extended (32b) format.	RO
28h	31:0	extended_minor	FW version's Minor field in extended (32b) format.	RO

Table 1404 - Firmware Info Field Descriptions

Offset	Bits	Name	Description	Access
2Ch	31:0	extended_sub_minor	FW version's SubMinor field in extended (32b) format.	RO
30h	15:0	isfu_major		RO
34h	2	sec_boot	0: Disable 1: Enable	RO
	1:0	life_cycle	0: Production 1: GA Secured 2: GA Non-Secured 3: RMA	RO

The numbers in the software section specify the oldest Driver version compatible with the current FW. See [Table 1405](#).

Table 1405 - Software Info Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0											00h
								major								minor								sub_minor								
rom0_arch				rom0_type				rom1_arch				rom1_type				rom2_arch				rom2_type				rom3_arch				rom3_type				
rom0_version (See Table 1407, “ROM Version Layout,” on page 1553)																												08h				
rom1_version (See Table 1407, “ROM Version Layout,” on page 1553)																												0Ch				
rom2_version (See Table 1407, “ROM Version Layout,” on page 1553)																												10h				
rom3_version (See Table 1407, “ROM Version Layout,” on page 1553)																												14h				
																												18h-1Ch				

Table 1406 - Software Info Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:16	major	Major Driver version number. [Internal] Not yet implemented. '0' - not supported.	RO
	15:8	minor	Minor Driver version number. [Internal] Not yet implemented. '0' - not supported.	RO
	7:0	sub_minor	Sub-minor Driver version number. [Internal] Not yet implemented. '0' - not supported.	RO

Table 1406 - Software Info Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:28	rom0_arch	Arch type of ROM 0: 0x0: unspecified 0x1: AMD64 - x86 64bit architecture 0x2: AARCH64 - ARM Architecture 64 bits 0x3: AMD64_AARCH64 - ROM code supporting both AMD64 and AARCH64 architectures 0x4: IA32 - Intel Architecture 32 bits	RO
	27:24	rom0_type	ROM 0 type: 0x0: none 0x1: Flexboot 0x2: UEFI 0x3: UEFI-CLP	RO
	23:20	rom1_arch	Arch type of ROM 1: 0x0: unspecified 0x1: AMD64 - x86 64bit architecture 0x2: AARCH64 - ARM Architecture 64 bits 0x3: AMD64_AARCH64 - ROM code supporting both AMD64 and AARCH64 architectures 0x4: IA32 - Intel Architecture 32 bits	RO
	19:16	rom1_type	ROM 1type: 0x0: none 0x1: Flexboot 0x2: UEFI 0x3: UEFI-CLP	RO
	15:12	rom2_arch	Arch type of ROM 2: 0x0: unspecified 0x1: AMD64 - x86 64bit architecture 0x2: AARCH64 - ARM Architecture 64 bits 0x3: AMD64_AARCH64 - ROM code supporting both AMD64 and AARCH64 architectures 0x4: IA32 - Intel Architecture 32 bits	RO
	11:8	rom2_type	ROM 2type: 0x0: none 0x1: Flexboot 0x2: UEFI 0x3: UEFI-CLP	RO
	7:4	rom3_arch	Arch type of ROM 3: 0x0: unspecified 0x1: AMD64 - x86 64bit architecture 0x2: AARCH64 - ARM Architecture 64 bits 0x3: AMD64_AARCH64 - ROM code supporting both AMD64 and AARCH64 architectures 0x4: IA32 - Intel Architecture 32 bits	RO
	3:0	rom3_type	ROM 3 type: 0x0: none 0x1: Flexboot 0x2: UEFI 0x3: UEFI-CLP	RO
		Mellanox Technologies		1552

Table 1406 - Software Info Field Descriptions

Offset	Bits	Name	Description	Access
08h	31:0	rom0_version (See Table 1407, “ROM Version Lay- out,” on page 1553)	ROM 0 version.	RO
0Ch	31:0	rom1_version (See Table 1407, “ROM Version Lay- out,” on page 1553)	ROM 1 version.	RO
10h	31:0	rom2_version (See Table 1407, “ROM Version Lay- out,” on page 1553)	ROM 2version.	RO
14h	31:0	rom3_version (See Table 1407, “ROM Version Lay- out,” on page 1553)	ROM 3 version.	RO

Table 1407 - ROM Version Layout

3	3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0										00h
major								minor								build															

Table 1408 - ROM Version Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	major	Major version	RO
	23:16	minor	Minor version	RO
	15:0	build	Build version	RO

Table 1409 - Development Info Layout

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	Offset
dev_branch_tag[0]								dev_branch_tag[1]								dev_branch_tag[2]								dev_branch_tag[3]								00h
dev_branch_tag[4]								dev_branch_tag[5]								dev_branch_tag[6]								dev_branch_tag[7]								04h
...																																...
dev_branch_tag[24]								dev_branch_tag[25]								dev_branch_tag[26]								dev_branch_tag[27]								18h

Table 1410 - Development Info Field Descriptions

Offset	Bits	Name	Description	Access
00h-18h	31:24	dev_branch_tag[28]	[Internal]: The tag for the dev branch should be <feature>_YYYY. The format of the string represented by ASCII.	RO

28.26.15MGPIR - Management General Peripheral Information Register

MGPIR register allows software to query the hardware and firmware general information of peripheral entities as Gearboxes etc.

Table 1411 - MGPIR - Management General Peripheral Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hw_info(see Table 1413, “Hardware Info Layout,” on page 1555)																															00h-1C	
reserved																															20h-9C	

Table 1412 - MGPIR - Management General Peripheral Information Register Fields

Offset	Bits	Name	Description	Access
00h-1Ch	256	hw_info(see Table 1413, “Hardware Info Layout,” on page 1555)	Hardware Information, see Table 1413, “Hardware Info Layout,” on page 1555	

Table 1413 - Hardware Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
				device_type				devices_per_flash																	num_of_devices								00h
																																04h	
																																08h-1Ch	

Table 1414 - Hardware Info Format

Offset	Bits	Name	Description	Access
00h	27:24	device_type	Device type. 0: No devices on system of that type. 1: Vega gearbox dice 2: Tiles 3: AGBM 4: XM	RO
	23:16	devices_per_flash	Number of devices of <i>device_type</i> per flash. [internal]: 0 for <i>device_type</i> Tiles. For device_type = Tiles, this field is irrelevant and set to '0'.	RO
	7:0	num_of_devices	Number of devices of <i>device_type</i> . [Internal]: In Tigris (SP-2), there are 32 Gearbox dice.	RO
04h	7:0	num_of_modules	Number of modules. [Internal]: Number of external modules.	RO

28.26.16MDIR - Management Device ID Register

This register is used to query the device ID and verify its genuineness.

Table 1415 - MDIR - Management Device ID Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
device_id																																00h-1Ch
																																20h-3Ch

Table 1416 - MDIR - Management Device ID Register Fields

Offset	Bits	Name	Description	Access
00h-1Ch	256	device_id	Device ID [Internal]: Calculated as SHA256(WAFER_INFO, X_COORDINATE, Y_COORDINATE)	RO

28.26.17 MSSIR - Management System Status Information Register

MSSIR register allows to query various system sticky bits and indications. The register supports up to 1024 sticky bits in total to monitor. [Internal]: This register is supported for Firebird system onwards.

Table 1417 - MSSIR - Management System Status Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												sys_idx	00h			
																													04h			
sys_info																													08h-84h			

Table 1418 - MSSIR - Management System Status Information Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	sys_idx	System Identifier 0: Firebird	Index
08h-84h	1024	sys_info	Firebird System Information, see Table 1401, “Hardware Info Layout,” on page 1546	

Table 1419 - MSSIR - Firebird Management System Status Information Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																									crc					00h		
																									bypass					04h		
																									fatal_dft					08h		
		freq5				freq4				freq3				freq2				freq1				freq0				0Ch						
																						freq7				freq6				10h		
																																14h-7Ch

Table 1420 - MSSIR - Management System Status Information Register Fields Description

Offset	Bits	Name	Description	Access
00h	7:0	crc	Each bit represent whether a CRC error occurred on the management interface to the tile number.	RO
04h	7:0	bypass	Each bit represent whether a bypass clock is run on the corresponding tile.	RO
08h	7:0	fatal_dft		RO
0Ch	29:25	freq5	Each field represent the frequency dividing factor for the core clock for each of the tiles	RO
	24:20	freq4		RO
	19:15	freq3		RO
	14:10	freq2		RO
	9:5	freq1		RO
	4:0	freq0		RO
10h	9:5	freq7		RO
	4:0	freq6		RO

28.26.18MHMPR - Management Host Management of Ports Register

Note: This register is not supported from VS-MAD access_reg.

The MHMPR register allows to set the status and query the status on the Host management of ports. In case it is not managed by Host, ports are able to rise by themselves and perform split auto-sensing.

Table 1421 - MHMPR - Management Host Management of Ports Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																upm	00h
																																	04h-10

Table 1422 - MHMPR - Management Host Management of Ports Register Field

Offset	Bits	Name	Description	Access
00h	0	upm	'0' - if the ports are managed by host. '1' - if the ports are not managed by host	RW

28.26.19MRTC - Management Real Time Clock

The management RTC allows the caller to sync the NIC Real Time Clock. When called, the NIC reserves the difference between its uptime and the provided time.

Table 1423 - MRTC - Management Real Time Clock Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
time_synced																																00h
																																04h
time_h																																08h
time_l																																0Ch

Table 1424 - MRTC - Management Real Time Clock Register Fields

Offset	Bits	Name	Description	Access
00h	31	time_synced	If set, the time returned in time_l/time_h is the number of microseconds passed since 1/1/1970. otherwise, the time_l/time_h measures the time since the NIC boot.	RO
08h	31:0	time_h	MSB of timestamp depending on time_synced	RW
0Ch	31:0	time_l	For more details about watchdog alert, see TBD. LSB of timestamp depending on time_synced	RW

28.26.20 MCWD – Management CPU Watchdog Register

[Internal]

MCWD Register is used to Configure the CPU watchdog mechanism.

Table 1425 - MCWD - Management CPU Watchdog Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
e																																00h
																																04h
																																08h

Table 1426 - MCWD - Management CPU Watchdog Register Fields

Offset	Bits	Name	Description	Access
00h	31	e	Enable watchdog (default disabled).	RW
	15:0	timer	Watchdog timer [msec]. When enabled, the SwitchX starts counting from the value written by the software to 0. If the SwitchX device reaches 0, the action configured is applied.	WO
04h	7:0	action	Action to apply in case the watchdog timer reaches 0. Bit 0 – send an alert. For more details about watchdog alert, see TBD. Bit 1 – set the Status LED to red. Bit 2 – reset the entire enclosure (This capability is only available in specific systems. For more details contact Mellanox support) Bit 3 – log in the firmware event log	RW

28.26.21MPFM – Management PCIe Failure Mode Register

MPFM register is used to query and update the PCIe Failure Mode.

Table 1427 - MPFM - Management PCIe Failure Mode Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															mode	00h
																																04h

Table 1428 - MPFM - Misc PCIe Failure Mode Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	mode	PCIe Failure Mode. 0 – normal. PCIe failure causes SwitchX reset. 1 – independent. PCIe failure does not cause SwitchX reset.	RW

28.26.22MRSR – Management Reset and Shutdown Register

MRSR register is used to reset or shutdown the switch or the entire system (when applicable).

Table 1429 - MRSR - Management Reset and Shutdown Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1430 - MRSR - Management Reset and Shutdown Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	command	Reset/shutdown command 0 – do nothing 1 – software reset 2 – hardware reset 3 – enclosure reset (requires system support) 4 – shutdown (requires system support)	WO

28.26.23MPGO - Management Pulse Generator Output

[Internal] TBD: global enable for the pps HW

TBD: lock/release

[Internal] TBD: pulse_duration default - AI EitanH ask Shlomi Tubul, currently written 100mSec

Table 1431 - MPGO Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
																															set_pd	00h		
																																04h		
next_pulse_time_stamp																															08h-0Ch			
																								pulse_duration										10h

Table 1432 - MPGO Register Field Description

Offset	Bits	Name	Description	Access
00h	31	lock	Write 0x1 to lock Pulse Generator During 'lock' operation, all other fields should be set to 0x0. TBD: we need a method for mutex between OS's	WO
00h	30	release	Write 0x1 to release Pulse Generator During 'release' operation, all other fields should be set to 0x0.	WO
00h	0	set_pd	Set pulse duration	WO
08h-0Ch	64	next_pulse_time_stamp	Next Pulse Time Stamp Start time of next trigger Units are device time units	RW
10h	9:0	pulse_duration	Pulse Duration Units are mSec Default is 100 [Internal] The clear should be done by FW in order to make sure that the GPIO is 0 Set only when <i>set_pd</i> = '1'	RW

28.26.24MLCR – Management LED Control Register

MLCR register is used to control the system LEDs.

Table 1433 - MLCR - Management LED Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_port								operation										operation cap	cap local or uid only	led_type				00h
																beacon_duration											04h					
																beacon_remain												08h				

Table 1434 - MLCR - Management LED Control Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	local_port	Local port number Not supported for CPU port (only supported when cap_local_or_uid_only==1.)	RW
	15:12	operation	0: Controlled by HW / Beacon 1: LED on - if bi-color, apply amber. 2: LED off Others: Reserved	RW
	5	operation_cap	If set to '1', operation field is supported.	RO
	4	cap_local_or_uid_only	If set, led_type of type UID and LOCAL Port only are supported.	RO
	3:0	led_type	Led type (only supported when cap_local_or_uid_only==1.) 0: LOCALP_AND_UID - local port and uid 1: UID 2: LOCALP - local port	RW
04h	15:0	beacon_duration	Duration of the beacon to be active, units of seconds The value of 0x0 will turn off the beacon The value of 0xffff will set the beacon to infinity	RW
08h	15:0	beacon_remain	Remaining duration of the beacon, units of seconds The value of 0xffff means infinity	RO

28.26.25MGCR - Management GPIO Capabilities Register

The register allows to monitor and control a list of supported General Purpose Input Output (GPIOs). This register can be used to access a vector which includes up to 32 signals at once. When a device has more than 32 GPIOs, then each group of (up to) 32 GPIOs will be accessed independently.

Table 1435 - MGCR - Management GPIO Capabilities Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																segments_count						segment						00h				
GPIO_data_in																															04h	
GPIO_data_out																															08h	
GPIO_set																															0Ch	
GPIO_clear																															10h	
GPIO_access_en																															14h	
direction																															18h	
drive_type																															1Ch	

Table 1436 - MGCR - Management GPIO Capabilities Register Layout

Offset	Bits	Name	Description	Access
00h	15:8	seg-ments_count	Ceiling function of total number of GPIOs / 32 in the system.	RO
	7:0	segment	The GPIO segment which the command's parameters apply	RW
04h	31:0	GPIO_data_in	Input value of accessible GPIO[i] signals. When GPIO_access_en[i] = 0, GPIO_data_in[i] always returns 0	RO
08h	31:0	GPIO_data_out	Output value of accessible GPIO signals. When GPIO_access_en[i] = 0, GPIO_data_out[i] always returns 0	RO
0Ch	31:0	GPIO_set	These 2 fields are used together to control GPIO signals in the following manner: GPIO_set[i]=0, GPIO_clear[i]=0 : GPIO_data_out[i] = No Change GPIO_set[i]=0, GPIO_clear[i]=1 : GPIO_data_out[i] = 0 GPIO_set[i]=1, GPIO_clear[i]=0 : GPIO_data_out[i] = 1 GPIO_set[i]=1, GPIO_clear[i]=1 : GPIO_data_out[i] = not(GPIO_data_out[i]) Ignored for non-controllable GPIO signals.	WO
10h	31:0	GPIO_clear		WO

Table 1436 - MGCR - Management GPIO Capabilities Register Layout

Offset	Bits	Name	Description	Access
14h	31:0	GPIO_access_en	Access allowance to the corresponding GPIO signal. 0: GPIO[i] is not observable / controllable by the host SW. 1: GPIO[i] is observable / controllable by the host SW.	RO
18h	31:0	direction	0: GPIO[i] is an input only signal 1: GPIO[i] is an output signal. Data out can be updates by the host SW and data in can be queried by the host SW. For non-accessible GPIOs, the corresponding value is 0.	RO
1Ch	31:0	drive_type	0: GPIO[i] is an open drain output. The device can only drive the GPIO[i] output pin low when GPIO_data_out[i] is 0. When GPI-O_data_out[i] is 1, GPIO[i] is in HighZ. 1: GPIO[i] is a full-drive output pin When GPIO_access_en[i] =0, direction[i] and drive_type[i] are always read as 0.	RO

28.26.26MERR - Management Entities Remapping Register

The register allows to remap the GPIOs assigned to an entity to another entity of the same type.

Table 1437 - MERR - Management Entities Remapping Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								current_index																ent_type				00h				
																								target_index				04h				
																																08h
																																0Ch
																																10h

Table 1438 - MERR - Management Entities Remapping Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	current_index	Current entity index	Index
	3:0	ent_type	Entity type. 0: Reserved 1: QSFP Module (the relevant signals are: present, reset, interrupt, power enable, low power mode, power good, modsel, QSFP LEDs)	Index
04h	7:0	target_index	Entity number to be mapped to.	RW

28.26.27MRMR – Management Resource Monitoring Register

MRSR register enables monitoring of internal device resources.

Table 1439 - MRMR - Management Resource Monitoring Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-0Ch
																																10-1Ch

Table 1440 - MRMR - Management Resource Monitoring Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	resource_id	Internal Resource Identifier 1 - Embedded TCAM other - reserved	Index
08h-0Ch	64	resource_state	Resource State The resource state encoding is per resource. TCAM: resource state [15:0] holds the number of used basic (18B) entries in the TCAM.	RO

28.27 Non-Volatile Registers

This section include the registers used for Non-Volatile configuration.

28.27.1 MNVDA - Non-Volatile Data Access

This register allows writing and reading non volatile configuration items. Supported Methods

Query - returns the Configuration Item header and the data according to the input *type* field.

Write - stores the Configuration Item in the Flash. If a Configuration Item with the given *type* and *type_mod* already exists, the new data replaces the previous one.

Return Status

Table 1441, “NVDA Register Return Status”

Table 1441 - NVDA Register Return Status

Status	Query	Write
ACCESS_OK	Operation succeeded	
ACCESS_BAD_PARAM	N.A.	Unknown domain or using a wrong type
ACCESS_RESOURCE_UNAVAILABLE	Configuration item not found	Not enough configuration area remaining
ACCESS_BAD_CONFIG	N.A.	Unsupported/invalid configuration
ACCESS_ERASE_EXCEEDED	N.A.	Flash erase counter exceeded specification limit
ACCESS_ERROR	Configuration section corrupted	
ACCESS_REG_LENGTH_TOO_SMALL	Register length is too short for the configuration item	N.A.

Table 1442 - MNVDA Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
configuration_item_header (See Table 1444, “Configuration Item Layout,” on page 1572)																															00h-08h	
configuration_item_data																															0Ch	

Table 1443 - MNVDA Register Field Description

Offset	Bits	Name	Description	Access
00h-08h	96	configuration_item_header (See Table 1444, “Configuration Item Layout,” on page 1572)		
0Ch	31:0	configuration_item_data		

Table 1444 - Configuration Item Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
valid					header_type				access_mode					writer_id				version				writer_host_id			length						00h	
type_class								type_index																				04h				
																crc16																08h

Table 1445 - Configuration Item Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:30	valid	[Internal] Encodes the data element validity 0x0: invalid_data- should be skipped 0x1: illegal- treated as Invalidated data element 0x2: valid_data 0x3: free- (no data exists)	RW
	27:26	header_type	[internal] Encodes the storage element Header Format 0x0: header_only - Only Header is protected by CRC 0x1: header_and_data - Header & Data are protected by CRC other values are reserved	RW

Table 1445 - Configuration Item Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:22	access_mode	Defines which value of the Configuration Item will be accessed. 0x0: NEXT - Next value to be applied 0x1: CURRENT - Currently set values (only valid for Query operation) Supported only if NVGC.nvda_read_current_settings==1. 0x2: FACTORY - Default factory values (only valid for Query operation). Supported only if NVGC.nvda_read_factory_settings==1.	Index

Table 1445 - Configuration Item Field Descriptions

Offset	Bits	Name	Description	Access
00h	20:16	writer_id	The entity which configured this parameter 0x0: UNSPECIFIED 0x1: CHASSIS_BMC 0x2: MAD 0x3: BMC 0x4: COMMAND_INTERFACE 0x5: ICMD - with unspecified source 0x6: ICMD_UEFI_HII - configured by the NIC's UEFI expansion ROM's HII menu. 0x7: ICMD_UEFI_CLP - configured by the NIC's expansion ROM's CLP. 0x8: ICMD_Flexboot - configured by the NIC's legacy expansion ROM. 0x9: ICMD_mlxconfig - configured by Mellanox mlxconfig tool 0xA: ICMD_USER1 - value available for customer created tools that uses the ICMD interface for writing TLVs. 0xB: ICMD_USER2 - value available for customer created tools that uses the ICMD interface for writing TLVs. 0xC: ICMD_MLXCONFIG_SET_RAW - configures by mlxconfig set raw operation. 0xD: ICMD_FLEXBOOT_CLP - configured by Legacy Expansion ROM CLP 0x10: BMC_APP1 - Configuration was done over the BMC by application #1 (application name is OEM specific) 0x11: BMC_APP2 - Configuration was done over the BMC by application #2 (application name is OEM specific) 0x12: BMC_APP3 - Configuration was done over the BMC by application #3 (application name is OEM specific) 0x1F: OTHER - the parameter was written by the NIC due to other reasons. Note - This field is writeable only when using the ICMD interface. The only value that are valid for writes are 0x6-0xB. Other values will be replaced by 0x5ICMD.	RO
	15:12	version	Configuration item version – defines the data structure following the header (together with the <i>type</i> field).	RW
	11:9	writer_host_id	This field identify the host that configured this TLV. The field is not relevant for the following writer_ids: unspecified, chassis BMC and MAD.	RO
	8:0	length	Length of configuration item data in bytes (not including header). Must be between 0 and 256.	RW

Table 1445 - Configuration Item Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:24	type_class	Type Class. 0x0: Global 0x1: Port 0x3: Host_PF 0x5: Log 0x6: File 0x7: Host	Index
	23:0	type_index	Configuration item index according to its type_class. Table 1446, “Configuration Item Type Class Global Layout,” on page 1575 Table 1448, “Configuration Item Type Class Physical Port Layout,” on page 1576 Table 1450, “Configuration Item Type Class Per Host-PF Layout,” on page 1576 Table 1452, “Configuration Item Type Class Log Layout,” on page 1576 Table 1454, “Configuration Item Type Class File Layout,” on page 1577 Table 1456, “Configuration Item Type Class Host Layout,” on page 1577	Index
08h	15:0	crc16	[Internal] Encodes the data integrity check of the data element. Note: In some elements, only encodes the header into the CRC16, as defined in the header type field.	RW

Table 1446 - Configuration Item Type Class Global Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class								parameter_index																							00h	

Table 1447 - Configuration Item Type Class Global Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x0	
	23:0	parameter_index		

Table 1448 - Configuration Item Type Class Physical Port Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class								port								parameter_index																00h

Table 1449 - Configuration Item Type Class Physical Port Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x1	
	23:16	port		
	15:0	parameter_index		

Table 1450 - Configuration Item Type Class Per Host-PF Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class								pf_index								pci_bus_index								parameter_index								00h

Table 1451 - Configuration Item Type Class Per Host-PF Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x3	
	23:18	pf_index		RW
	17:10	pci_bus_index		RW
	9:0	parameter_index		

Table 1452 - Configuration Item Type Class Log Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class								parameter_index																							00h	

Table 1453 - Configuration Item Type Class Log Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x5	
	23:0	parameter_index		

Table 1454 - Configuration Item Type Class File Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class								parameter_index																							00h	

Table 1455 - Configuration Item Type Class File Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x6	
	23:0	parameter_index		

Table 1456 - Configuration Item Type Class Host Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type_class														parameter_index																	00h	

Table 1457 - Configuration Item Type Class Host Field Description

Offset	Bits	Name	Description	Access
00h	31:24	type_class	type_class = 0x7	
	17:0	parameter_index		

28.27.2 MNVDI - Non-Volatile Data Invalidate

This register allows invalidating non-volatile configuration items. Register Format

[Table 1444, “Configuration Item Layout”](#).

Supported Methods

Write - The Configuration Item is invalidated.

Return Status

[Table 1460, “NVDI Register Return Status”](#)

Table 1460 - NVDI Register Return Status

Status	Query	Write
ACCESS_OK	N.A.	Operation succeeded
ACCESS_NOT_SUPP_METHOD	Query is not supported for this register	N.A.
ACCESS_RESOURCE_UNAVAIL- ABLE	N.A.	Configuration item not found

Table 1461 - MNVDI Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
configuration_item_header (Table 1444, “Configuration Item Layout,” on page 1572)																															00h-08h	

Table 1462 - MNVDI Register Field Description

Offset	Bits	Name	Description	Access
00h-08h	96	configuration_item_header (Table 1444, “Configuration Item Layout,” on page 1572)		

28.27.3 MNVIA - Non-volatile data Invalidate All

This register invalidates all non-volatile settings. It invalidates TLVs that belong to the used function and host. For instance, in a multi-host system, this register may invalidate all per-host, per-function settings belonging to the function being used, while in a single-host system it may also invalidate physical port settings belonging to the function's physical port.

Supported Methods

Write - All the Configuration Items are invalidated.

Return Status

Table 1463, "NVIA Register Return Status"

Table 1463 - NVIA Register Return Status

Status	Query	Write
ACCESS_OK	N.A.	Operation succeeded
ACCESS_NOT_SUPP_METHOD	Query is not supported for this register	N.A.

Table 1464 - MNVIA Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																							writer_id				target			00h		
																															04h	

Table 1465 - MNVIA Register Field Description

Offset	Bits	Name	Description	Access
00h	8:4	writer_id	The entity which perform the invalidate. The encoding same as writer_id in Configuration Item register (See Table 1444 , "Configuration Item Layout," on page 1572).	
	2:0	target	Information targeted for invalidation 0: ALL - all NVRAM parameters. [Internal] except NVLOG 1: [Internal] NVLOG All other values are reserved.	WO

28.27.4 MNVGC - Non-volatile Global Capabilities

The NVGC command is used to query global capabilities of the NV infrastructure (that is, capabilities that are not unique for a specific type).

Table 1466 - MNVGC Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h-0Ch

Table 1467 - MNVGC Register Field Description

Offset	Bits	Name	Description	Access
00h	2	priv_nv_other_host	When set, TLVs of other hosts may be modified.	RO
	1	nvda_read_current_settings	NVDA Read current settings. Indicates if reading the current settings by NVDA is supported. 0: unsupported 1: supported	RO
	0	nvda_read_factory_settings	NVDA Read factory settings. Indicates if reading the factory settings by NVDA is supported. 0: unsupported 1: supported	RO

28.27.5 MNVGN - Non-Volatile Get Next

This register allows systematic reading of all configured nonvolatile parameters one by one. To initiate the operation, the caller of the NVGN command sets the pointer to 0x0, the returned value is first valid NVRAM parameter together with the a new pointer. To get the 2nd parameter, the caller uses the new pointer and so on. If there are no more parameters to read, the NVGN command returns a 0x0 pointer.

The caller can assume nothing about the returned pointer and pointer arithmetic is not allowed.

The function only returns parameters that are applicable for the caller. Especially, a caller running on host H that uses the command interface or ICMD over PCI function F, will only receive type_class=3 (type_per_host_function) parameters that were configured by host H through function F.

Return Status

Table 1468, “NVGN Register Return Status”

Table 1468 - NVGN Register Return Status

Status	Query
ACCESS_OK	Operation succeeded
ACCESS_RESOURCE_UNAVAILABLE	The provided NVGN pointers belongs to the non-active NVconfig bank (Typically, this may happen if garbage collection was performed since the last NVGN command).

Table 1469 - MNVGN Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nv_pointer																																00h
																																04h-0Ch
configuration_item_header (See Table 1444, “Configuration Item Layout,” on page 1572)																																10h-18h
configuration_item_data																																1Ch-..

Table 1470 - MNVGN Register Field Description

Offset	Bits	Name	Description	Access
00h	31:0	nv_pointer	Pointer to the NV parameter (see description above for more information)	Index
10h-18h	96	configuration_item_header (See Table 1444, “Configuration Item Layout,” on page 1572)		RO
1Ch-..	31:0	configuration_item_data		RO

28.27.6 MGNLE - Management Get Next Log Entry

This register allows systematic reading of all internal log entries one by one. To initiate the operation, the caller of the MGNLE command sets the pointer to 0x0, the returned value is first log entry with the a new pointer. To get the 2nd entry, the caller uses the new pointer and so on. If there are no more entries to read, the MGNLE command returns a 0x0 pointer.

Table 1471 - MGNLE Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
le_pointer																																00h	
synced time				lost_events																													04h
	time_h																																08h
	time_l																																0Ch
	configuration_item_header (See Table 1444. “Configuration Item Layout,” on page 1572)																																10h-18h
	log_data																																1Ch-...

Table 1472 - MGNLE Register Field Description

Offset	Bits	Name	Description	Access
00h	31:0	le_pointer	Pointer to the log entry (see description above for more information)	Index
04h	31	synced_time	0x0: Boot - time is measured since NIC bootup. 0x1: Synced - time was synced by the host.	RO
	27:24	lost_events	The number of events that the system failed to log since the previous log entry. 15 should be interpreted as '15 or more events were lost'	RO
08h	31:0	time_h	The 32 MSB of the 64bit timestamp. In microseconds When 'time_synced' bit is set this is the time passed since the 1/1/1970 epoch. When 'time_synced' is cleared this is the NIC uptime.	RO
0Ch	31:0	time_l	The 32 LSB of the 64bit timestamp. In microseconds	RO

Table 1472 - MGNLE Register Field Description

Offset	Bits	Name	Description	Access
10h-18h	96	configuration_item_header (See Table 1444, “Configuration Item Layout,” on page 1572)		RO
1Ch-..	31:0	log_data	The logged data. Format is different per the according to the log type.	RO

28.27.7 MNVCT - Non-volatile commit

[Internal and not implemented]

This register applies Configuration Items setting done by NVDA register with the *shadow* field set. See “MNVDA - Non-Volatile Data Access” on page 1571

Register Format

Zero-length register.

Supported Methods

Write - All the Configuration Items done by NVDA register with the *shadow* field set, are stored in Flash.

Return Status

Table 1474, “NVQC Register Return Status”

Table 1473 - NVCT Register Return Status

Status	Query	Write
ACCESS_OK	N.A.	Operation succeeded
ACCESS_NOT_SUPP_METHOD	Query is not supported for this register	N.A.

28.27.8 MNVQC - Non-volatile query capabilities

This register exposes the supported set of configuration items. The register is applicable only for Configuration Items targeted at the device.

Supported Methods

Write - All the Configuration Items done by NVDA register with the *shadow* field set, are stored in Flash.

Return Status

Table 1474, “NVQC Register Return Status”

Table 1474 - NVQC Register Return Status

Status	Query	Write
ACCESS_OK	N.A.	Operation succeeded
ACCESS_NOT_SUPP_METHOD	Query is not supported for this register	N.A.

Table 1475 - MNVQC Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
type																																00h
																								version				support_wr	support_rd	04h		

Table 1476 - MNVQC Register Field Description

Offset	Bits	Name	Description	Access
00h	31:0	type	Configuration item type. Table 1446, “Configuration Item Type Class Global Layout,” on page 1575 Table 1448, “Configuration Item Type Class Physical Port Layout,” on page 1576 Table 1450, “Configuration Item Type Class Per Host-PF Layout,” on page 1576	Index

Table 1476 - MNVQC Register Field Description

Offset	Bits	Name	Description	Access
04h	7:4	version	The maximal version of the configuration item supported by the device	RO
	1	support_wr	The configuration item is supported and can be updated	RO
	0	support_rd	The configuration item is supported and can be read	RO

28.28 Tracer Registers

This section include the registers used to configure and enable Tracer.

Table 1477, “Tracer Registers Summary,” on page 1587 summarize Tracer Registers.

Table 1477 - Tracer Registers Summary

Register Name	Register ID (hex)	Query	Write	Event	Init/ runtime	Reference
MTRC_CAP	0x9040	v				See Table 1478, “MTRC_CAP - Tracer Capability Register Layout,” on page 1587
MTRC_CONF	0x9041	v	v			See Table 1482, “MTRC_CONF - Tracer Configuration Register Layout,” on page 1589
MTRC_STDB	0x9042					See Table 1484, “MTRC_STDB - Tracer String DB Register Layout,” on page 1590
MTRC_CTRL	0x9043	v	v			See Table 1486, “MTRC_CTRL - Tracer Control Register Layout,” on page 1591

28.28.1 MTRC_CAP -Tracer Capability Register

The support for this register is indicated by MCAM.mng_access_reg_cap_mask.

Table 1478 - MTRC_CAP - Tracer Capability Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
trace_owner	trace_to_memory					trc_ver																					num_string_db				00h	
		first_string_trace						num_string_trace																								04h

Table 1478 - MTRC_CAP - Tracer Capability Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								log_max_trace_buffer_size				08h				
																																0Ch
string_db_param[0]																																10h-14h
string_db_param[1]																																18h-1Ch
...																																...
string_db_param[7]																																48h-4Ch
																																50h-80h

Table 1479 - MTRC_CAP - Tracer Capability Register Fields

Offset	Bits	Name	Description	Access
00h	31	trace_owner	Write 0x1 to register for tracer ownership, write 0x0 to de-register. Read value 0x1 indicates tracer ownership is granted	RW
	30	trace_to_memory	When set the device supports logging traces to memory	RO
	25:24	trc_ver	Indicates the version of the tracing mechanism. See, “Timestamp Event Traces” on page 1053 0x0: VER_0 0x1: VER_1 other values are reserved	RO
	3:0	num_string_db	Number of different string sections building the database	RO
04h	31:24	first_string_trace	The lowest <i>event_id</i> that should be interpreted as a string trace	RO
	23:16	num_string_trace	The number of consecutive <i>event_id</i> that should be interpreted as a string trace	RO
08h	7:0	log_max_trace_buffer_size	Log 2 of the maximal size of the trace buffer given in units of 4KB	RO

Table 1479 - MTRC_CAP - Tracer Capability Register Fields

Offset	Bits	Name	Description	Access
10h-4Ch	64	string_d-b_param[8] (See Table 1480, “String DB Section Parameters Layout,” on page 1589)	String DB section parameters.	RO

Table 1480 - String DB Section Parameters Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
string_db_base_address																															00h	
								string_db_size																							04h	

Table 1481 - String DB Section Parameters Description

Offset	Bits	Name	Description	Access
00h	31:0	string_d-b_base_address	Offset of the first string of the section, relative to the entire string data base, given in bytes.	RO
04h	23:0	string_db_size	Size of string database section, given in bytes	RO

28.28.2 MTRC_CONF - Tracer Configuration Register

This register configures the Tracer. It is supported only if HW tracer is supported - TRC_CAP.
dump_mode_mask != 0.

Table 1482 - MTRC_CONF - Tracer Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
																																trace_mode	00h
																								log_trace_buffer_size								04h	
trace_mkey																																08h	
																																0Ch-7Ch	

Table 1483 - MTRC_CONF - Tracer Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	trace_mode	Tracing mode 0x1: TRACE_TO_MEMORY Other values are reserved.	RW
04h	7:0	log_trace_buffer_size	Log 2 of the Size of the trace buffer, given in units of 4KB. Value should not exceed log_max_trace_buffer_size. Valid only for trace_mode TRACE_TO_MEMORY. Modifying this parameter after the tracer was active may cause loss of sync regarding the location of the next trace.	RW
08h	31:0	trace_mkey	MKey registered for the trace buffer access. Valid only for <i>trace_mode</i> TRACE_TO_MEMORY. Modifying this parameter after the tracer was active may cause loss of sync regarding the location of the next trace.	RW

28.28.3 MTRC_STDB - Tracer String DB Register

Table 1484 - MTRC_STDB - Tracer String DB Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
string_d- b_index								read_size																				00h				
start_offset																															04h	
string_db_data[0]																															08h	
string_db_data[1]																															0Ch	
...																															...	

Table 1485 - MTRC_STDB - Tracer String DB Register Fields

Offset	Bits	Name	Description	Access
00h	31:28	string_db_index	The section of the String DB being accessed	Index
	23:0	read_size	The number of bytes to read from the String DB. The number of bytes must: 1. Be a multiple of 64B 2. Not exceed the String DB section (with <i>start_offset</i>) 3. Not exceed the limitations defined by the medium carrying the Register.	Index

Table 1485 - MTRC_STDB - Tracer String DB Register Fields

Offset	Bits	Name	Description	Access
04h	31:0	start_offset	The offset in the String DB to read from, given in Bytes. The offset must: 1. Be a multiple of 64B 2. Not exceed the String DB section (with <i>read_size</i>)	Index
08h-...	31:0	string_db_data[...]	Strings from the database. The size of this field is defined by <i>read_size</i>	RO

28.28.4 MTRC_CTRL - Tracer Control Register

Table 1486 - MTRC_CTRL - Tracer Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
trace_status				arm_event												modify_field_select															00h	
																																04h
											current_timestamp[52:32]															08h						
current_timestamp[31:0]																																0Ch
																																10h-3Ch

Table 1487 - MTRC_CTRL - Tracer Control Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	trace_status	Current status of the tracer 0x0: DISABLED - logging traces is stopped 0x1: ACTIVE - logging traces is active	RW
	27	arm_event	When set, an event will be generated if new Tracer events were logged since last event.	WO
	15:0	modify_field_select	Field select indicated which writable fields to modify bit 0: trace_status Other bits are reserved.	WO
08h	20:0	current_timestamp[52:32]	MSB of the current timestamp counter	RO

Table 1487 - MTRC_CTRL - Tracer Control Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:0	current_time-stamp[31:0]	LSB of the current timesatmp counter	RO

28.29 PCIe Registers

28.29.1 MPEIN - Management PCIe INfo Register

The MPEIN register provides PCIe specific information.

Table 1488 - MPEIN - Management PCIe INfo Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset							
		depth							pcie_index							node																	00h						
capability_mask																															04h								
								link_width_enabled							link_speed_enabled														08h										
lane0_physical_position								link_width_active							link_speed_active														0Ch										
num_of_pfs															num_of_vfs																10h								
bdf0																																							14h
max-read_re-quest_size				max_pay-load_size										pwr_sta-tus		port_type																	lane reversal	18h					
																				pci_power								1Ch											
																																	20h						
device_status															port_state																		24h						
																receiver_detect_result																28h							
																																	2Ch						

Table 1489 - MPEIN - Management PCIe INfo Register Fields

Offset	Bits	Name	Description	Access
00h	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	PCIe index number (Internal domain index) Reserved when access is from the host, but can be used when operating in Socket-Direct mode.	Index
	15:8	node	the node within each depth.	Index
04h	31:0	capability_mask	[Internal] TBD	RO

Table 1489 - MPEIN - Management PCIE INfo Register Fields

Offset	Bits	Name	Description	Access
08h	23:16	link_width_enabled	Maximum Link Width enabled: 0: 1x 1: 2x 2: 4x 3: 8x 4: 16x	RO
	15:0	link_speed_enabled	Max Link Speed: Bit 0: 2.5G - (Gen1) Bit 1: 5G - (Gen2) Bit 2: 8G - (Gen3) Bit 4: 16G - (Gen4)	RO
0Ch	31:24	lane0_physical_position	The physical lane position of logical lane0	RO
	23:16	link_width_active	Negotiated Link Width, pcie_link_width active: 0 - 1x 1 - 2x 2 - 4x 3 - 8x 4 - 16x	RO
	15:0	link_speed_active	Current Link Speed: Bit 0 - 2.5G (Gen1) Bit 1 - 5G (Gen2) Bit 2 - 8G (Gen3) Bit 4- 16G (Gen4)	RO
10h	31:16	num_of_pfs	Number of Physical Functions (PFs)	RO
	15:0	num_of_vfs	Number of Total Virtual Functions (for all PFs)	RO
14h	31:16	bdf0	Bus Device Function - only for function0	RO

Table 1489 - MPEIN - Management PCIE Info Register Fields

Offset	Bits	Name	Description	Access
18h	31:28	max_read_request_size	Max read request size in bytes: 0 - 128B 1 - 256B 2 - 512B 3 - 1024B 4 - 2048B 5 - 4096B	RO
	27:24	max_payload_size	Max payload size in bytes: 0 - 128B 1 - 256B 2 - 512B 3 - 1024B 4 - 2048B 5 - 4096B	RO
	18:16	pwr_status	Indicates the status of PCI power consumption limitations. 0: PCI power report could not be read. 1: Sufficient power reported. 2: Insufficient power reported. 3-7: Reserved	RO
	15:12	port_type	Indicates the specific type of this PCI Express Function. Note that different Functions in a multi-Function device can generally be of different types. 0 - PCI Express Endpoint port 4 - Root Port of PCI Express Root Complex 5 - PCI Express Upstream port 6 - PCI Express Downstream port	RO
	0	lane_reversal	Reversal mode of the link: 0 - straight 1 - reversal Note: together with <i>lane0_physical_position</i> provide the physical lane.	RO
1Ch	11:0	pci_power	Power reported by the PCI device. The units are in Watts. 0: Power is unknown.	RO
24h	31:16	device_status	device_status bit mask: Bit 0: Correctable_error Bit 1: Non_Fatal_Error_detection Bit 2: Fatal_Error_detected Bit 3: Unsupported_request_detected Bit 4: AUX_power Bit 5: Transaction_Pending	RO

Table 1489 - MPEIN - Management PCIE Info Register Fields

Offset	Bits	Name	Description	Access
24h	15:8	port_state	[Internal] Indicates the status of the PCIE state machine: 0x00 - Disable TS1 0x01 - Disable Ack 0x02 - Disable Wait 0x10 - RESET 0x20 - Loop Back Master Active 0x21 - Loop Back Master Exit 0x31 - Loop Back Slave Entry 0x32 - Loop Back Slave Align 0x33 - Loop Back Slave EB Threshold 0x34 - Loop Back Slave Active 0x35 - Loop Back Slave Idle 0x36 - Loop Back Slave Exit 0x40 - Detect Quiet 0x41 - Detect Active 1 0x42 - Detect Active 2 0x50 - Polling Active 0x51 - Polling Compliance Entry 0x52 - Polling Compliance 0x53 - Polling Compliance Exit 0x54 - Polling Config 0x60 - Config Width Start 0x61 - Config Width Ack 0x62 - Config Lane Wait 0x63 - Config Lane Ack 0x64 - Config Complete 0x65 - Config Idle 0x70 - L0_UP 0x71:0x7F - L0s 0x80 - L1 ENTER 0x81 - L1 ACTIVE 0x90 - L1 EXIT 0xA0 - Recover Receiver Lock 0xA1 - Recover Speed 0xA2 - Recover Config 0xA2 - Recover Idle 0xA5 - Recover ESpeed 0xA8 - Recover Equalization Phase 0 0xA9 - Recover Equalization Phase 1 0xAA - Recover Equalization Phase 2 0xAB - Recover Equalization Phase 3 0xAC - Recover Equalization Idle 0xB0 - Force TS1 0xC0 - Force TS2 0xD0 - Force IDLE 0xE0 - Force COMP 0xF0 - Force disable	RO

Table 1489 - MPEIN - Management PCIe Info Register Fields

Offset	Bits	Name	Description	Access
28h	15:0	receiver_detect_result	[Internal] receiver detect result bit mask per lane	RO

28.29.2 MPCNT - Management PCIe Counters Register

The MPCNT register retrieves PCIe counters.

Table 1490 - MPCNT - Management PCIe Counters Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-FCh

Table 1491 - MPCNT - Management PCIe Counters Register Fields

Offset	Bits	Name	Description	Access
00h	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	PCIe index number (Internal domain index) Reserved when access is from the host	Index
	15:8	node	the node within each depth.	Index
	5:0	grp	Performance counter group. [Internal] Group 63 indicates all groups. Only valid on Set() operation with <i>clr</i> bit set. 0: PCIE_performance_counters 1: PCIE_lane_counters - [Internal] 2: PCIE_timers_and_states - [Internal] [Internal] NOTE: group 2 counters can't be cleared (also when using group 63)	Index
04h	31	clr	Clear counters. Setting the <i>clr</i> bit will reset the counter value for all counters in the counter group. This bit can be set for both Set() and Get() operation.	OPWO

Table 1493 - PCIe Performance Counters Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	life_time_counter_high	[Internal] Life time counter - can't be cleared.	RO
04h	31:0	life_time_counter_low		RO
08h	31:0	rx_errors	Number of transitions to recovery due to Framing errors and CRC (dlp and tlp) errors.	RO
0Ch	31:0	tx_errors	Number of transitions to recovery due to EIEOS and TS errors.	RO
10h	31:0	l0_to_recovery_eieos	Number of transitions to recovery due to getting EIEOS	RO
14h	31:0	l0_to_recovery_ts	Number of transitions to recovery due to getting TS	RO
18h	31:0	l0_to_recovery_framing	Number of transitions to recovery due to identifying framing errors at gen3/4.	RO
1Ch	31:0	l0_to_recovery_retrain	Number of transitions to recovery due to link retrain request from data link	RO
20h	31:0	crc_error_dlp	Number of transitions to recovery due to identifying CRC DLLP errors	RO
24h	31:0	crc_error_tlp	Number of transitions to recovery due to identifying CRC TLP errors	RO
28h-2Ch	64	tx_overflow_buffer_pkt	The number of packets dropped due to lack of PCIe buffers or receive path from NIC port toward the hosts.	RO
30h	31:0	outbound_stalled_reads	The percentage of time within the last second that the NIC had outbound non-posted read requests but could not perform the operation due to insufficient non-posted credits. Supported only when MCAM.pcie_outbound_stalled == 1.	RO
34h	31:0	outbound_stalled_writes	The percentage of time within the last second that the NIC had outbound posted writes requests but could not perform the operation due to insufficient posted credits. Supported only when MCAM.pcie_outbound_stalled == 1.	RO

Table 1493 - PCIe Performance Counters Group Fields

Offset	Bits	Name	Description	Access
38h	31:0	outbound_stalled_reads_events	The number of events where outbound_stalled_reads was above a threshold. The threshold is configured by outbound_stalled_reads_th in MPEGC. Supported only when MCAM.pcie_outbound_stalled ==1.	RO
3Ch	31:0	outbound_stalled_writes_events	The number of events where outbound_stalled_writes was above a threshold. The threshold is configured by outbound_stalled_writes_th in MPEGC. Supported only when MCAM.pcie_outbound_stalled ==1.	RO
40h-44h	64	tx_overflow_buffer_marked_pkt	The number of packets marked due to lack of PCIe buffers or receive path from NIC port toward the hosts. Supported only if MCAM.mark_tx_action_cqe or MCAM.mark_tx_action_cnp are set to '1'.	RO

28.29.2.2 [Internal] PCIE timers and states

Table 1494 - PCIE timers and states Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
life_time_counter_high																															00h	
life_time_counter_low																															04h	
time_to_boot_image_start																															08h	
time_to_link_image																															0C <h></h>	
calibration_time																															10h	
time_to_first_perst																															14h	
time_to_detect_state																															18h	
time_to_l0																															1Ch	
time_to_crs_en																															20h	
time_to_plastic_image_start																															24h	
time_to_iron_image_start																															28h	
perst_handler																															2Ch	
times_in_l1																															30h	
times_in_l23																															34h	

Table 1494 - PCIE timers and states Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dl_down																																38h
config_cycle1usec																																3Ch
config_cycle2to7usec																																40h
config_cycle8to15usec																																44h
config_cycle16to63usec																																48h
config_cycle64usec																																4Ch
correctable_err_msg_sent																																50h
non_fatal_err_msg_sent																																54h
fatal_err_msg_sent																																58h
																																5C-F4h

Table 1495 - PCIE timers and states Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	life_time_counter_high	[Internal] Life time counter - can't be cleared.	RO
04h	31:0	life_time_counter_low		RO
08h	31:0	time_to_boot_image_start	Time form start until FW boot image starts running in usec. Note: can't be cleared.	RO
0Ch	31:0	time_to_link_image	Time form start until FW pci_link image starts running in usec.	RO
10h	31:0	calibration_time	Time it took FW to do calibration in usec.	RO
14h	31:0	time_to_first_perst	Time form start until FW handle first prest	RO
18h	31:0	time_to_detect_state	Time form start until first transition to LTSSM.Detect_Q	RO
1Ch	31:0	time_to_l0	Time form start until first transition to LTSSM.L0	RO
20h	31:0	time_to_crs_en	Time form start until crs is enabled	RO
24h	31:0	time_to_plastic_image_start	Time form start until FW plastic image starts running in usec.	RO
28h	31:0	time_to_iron_image_start	Time form start until FW iron image starts running in usec.	RO
2Ch	31:0	perst_handler	Number of persts arrived	RO
30h	31:0	times_in_l1	Number of times LTSSM entered L1 flow	RO

Table 1496 - PCIE lanes counters Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
error_counter_lane11																															34h	
error_counter_lane12																															38h	
error_counter_lane13																															3Ch	
error_counter_lane14																															40h	
error_counter_lane15																															44h	
																															48-4Fh	

Table 1497 - PCIE lanes counter Group Fields

Offset	Bits	Name	Description	Access
00h	31:0	life_time_counter_high	[Internal] Life time counter - can't be cleared.	RO
04h	31:0	life_time_counter_low		RO
08h	31:0	error_counter_lane0	Number of errors detected on logical lane 0	RO
0Ch	31:0	error_counter_lane1	Number of errors detected on logical lane 1	RO
10h	31:0	error_counter_lane2	Number of errors detected on logical lane 2	RO
14h	31:0	error_counter_lane3	Number of errors detected on logical lane 3	RO
18h	31:0	error_counter_lane4	Number of errors detected on logical lane 4	RO
1Ch	31:0	error_counter_lane5	Number of errors detected on logical lane 5	RO
20h	31:0	error_counter_lane6	Number of errors detected on logical lane 6	RO
24h	31:0	error_counter_lane7	Number of errors detected on logical lane 7	RO
28h	31:0	error_counter_lane8	Number of errors detected on logical lane 8	RO
2Ch	31:0	error_counter_lane9	Number of errors detected on logical lane 9	RO
30h	31:0	error_counter_lane10	Number of errors detected on logical lane 10	RO
34h	31:0	error_counter_lane11	Number of errors detected on logical lane 11	RO
38h	31:0	error_counter_lane12	Number of errors detected on logical lane 12	RO
3Ch	31:0	error_counter_lane13	Number of errors detected on logical lane 13	RO
40h	31:0	error_counter_lane14	Number of errors detected on logical lane 14	RO
44h	31:0	error_counter_lane15	Number of errors detected on logical lane 15	RO

28.29.3 MPEGC - Management PCIe General Configuration Register

The MPEBC register is used to configure and access to PCIe general configuration.

For HCA device, supported only if reported in MCAM register.

Table 1498 - MPEGC - Management PCIe General Configuration Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
		depth						pcie_index						node														00h									
																field_select																		04h			
																tx_lossless_overflow_oper																		tx_lossy_overflow_oper		08h	
																																		mark_cnp		tx_overflow_sense	0Ch
																																		mark_cqe			
																out-bound_stalled_reads_th								out-bound_stalled_writes_th								10h-28h					

Table 1499 - MPEGC - Management PCIe General Configuration Register Fields

Offset	Bits	Name	Description	Access
00h	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	PCIe index number (Internal domain index) Reserved when access is from the host	Index
	15:8	node	the node within each depth.	Index
04h	15:0	field_select	Field select indicated which writable fields to modify. Bit 0: tx_lossy_overflow_oper Bit 1 : outbound_stalled_reads_th Bit 2 :outbound_stalled_writes_th Bit 3 : tx_overflow_sense Bit 4 : mark_tx_action_cqe Bit 5 : mark_tx_action_cnp	WO

Table 1499 - MPEGC - Management PCIE General Configuration Register Fields

Offset	Bits	Name	Description	Access
08h	31	tx_overflow_sense	Set the sensibility level of the tx overflow mechanism. 0 - Aggressive [Internal]: WRED is on. 1 - Dynamic adjustment. When tx_lossy_overflow_oper is disabled, tx_overflow_sense must be disabled. [Internal]: Based on RX buffer fullness. Supported only when MCAM.dynamic_tx_overflow is '1'.	RW
	30	mark_cqe	When this bit is set, the marking action will be set in the CqE for TCP traffic. Supported only when MCAM.mark_tx_action_cqe is '1'.	RW
	29	mark_cnp	When this bit is set, the marking action will be generating a CNP for RoCE traffic. Supported only when MCAM.mark_tx_action_cnp is '1'.	RW
	9:8	tx_lossless_overflow_oper	[Internal]: When overflow encountered for lossless packets, it will be marked and will be counted in "tx_overflow_buffer_marked_pkt" counter. 00 - Disabled 10 - mark - overflow traffic will be marked in the CE field in the CqE. Mark can be set only if tx_lossy_overflow_oper is set to disable or mark. Supported only when MCAM.mark_tx_action_cqe or MCAM.mark_tx_action_cnp are '1'. [internal] : In CX4 and CX5 the configuration is global and last set is used.	RW
	1:0	tx_lossy_overflow_oper	When overflow encountered for lossy packets, it will be dropped or marked and will be counted in "tx_overflow_buffer_dropped_pkt" or "tx_overflow_buffer_marked_pkt" counter. 00 - Disabled 01 - drop - overflow traffic will be dropped. [Internal]: Drop can be set only if tx_lossless_overflow_oper is set to disable. backward enabled. 10 - mark - overflow traffic will be marked in the CE field in the CqE. [Internal]: Mark can be set only if tx_lossless_overflow_oper is set to disable or mark. Supported only when MCAM.mark_tx_action_cqe or MCAM.mark_tx_action_cnp are '1'. [internal] : In CX4 and CX5 the configuration is global and last set is used.	RW
0Ch	14:8	out-bound_stalled_reads_th	Each time that the counter outbound_stalled_reads will exceed this threshold, will be counted in the counter out-bound_stalled_reads_events – range 0...100.	RW
	6:0	out-bound_stalled_writes_th	Each time that the counter outbound_stalled_writes will exceed this threshold, will be counted in the counter out-bound_stalled_writes_events – range 0...100.	RW

28.29.4 MPIR - Management PCIe Index Register

The register allows to query the PCIe indexes.

Table 1500 - MPIR - Management PCIe Index Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sdm			depth					pcie_index					node								host_buses			00h								
																secondary_bus					subordinate_bus									04h		
local_port								bus										device						08h								
slot_cap								num_con_devices							slot_number										0Ch							

Table 1501 - MPIR- Management PCIe Index Register Fields

Offset	Bits	Name	Description	Access
00h	31	sdm	Socket-Direct mode indication. 0: non-Socket-Direct mode (single host or multi-host) 1: Socket-Direct mode, for querying host	RO
	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	Internal domain index	Index
	15:8	node	the node within each depth.	Index
	3:0	host_buses	Number of PCIe buses available for the host to connect ot the device. '0' when operating in non-Socket-Direct mode.	RO
04h	15:8	secondary_-bus	secondary bus - the internal logic bus in the switch. Default value of '0' in case it is not a switch port.	RO
	7:0	subordinate_-bus	sunbordinate bus - the highest bus number that subordinates to switch. Default value of '0' in case it is not a switch port.	RO
08h	31:24	local_port	Local port number	RO
	23:16	bus	PCIe bus number.	RO
	7:3	device	PCIe device number.	RO
0Ch	31	slot_cap	If set to '1', <i>slot_number</i> field is supported.	RO
	23:16	num_con_de-vices	number of PCIe connected deices / EP on the current port.	RO
	12:0	slot_number	Slot number	RO

28.29.5 MPECS - Management PCIE Configuration Space Register

The MPECS register is used to configure and access to PCIE configuration space. The specific configuration space registers are defined in the PCIE spec.

Table 1502 - MPECS - Management PCIE Configuration Space Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
		depth							pcie_index							node														00h		
																capability_id																04h
																reg_num																08h
								func_num							status											byte_enable				0Ch		
																															ext	10h
data																																14h
																																18h-1Ch

Table 1503 - MPECS - Management PCIE Configuration Space Register Fields

Offset	Bits	Name	Description	Access
00h	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	PCIE index number Reserved when access is from the host	Index
	15:8	node	the node within each depth.	Index
04h	15:0	capability_id	capability ID as defined in PCIE spec	Index
08h	15:0	reg_num	Register number Byte offset from the beginning of the capability ID	Index

Table 1503 - MPECS - Management PCIE Configuration Space Register Fields

Offset	Bits	Name	Description	Access
0Ch	23:16	func_num	Function number physical function index - see PCIE spec	Index
	15:8	status	PCIE configuration space access status: 0x0: SUCCESS - successful access 0x1: FUNC_NUM_FAILURE - accessing unsupported configuration space function number 0x2: CAP_ID_FAILURE - accessing unsupported configuration space capability ID number 0x3: REG_NUM_FAILURE - accessing unsupported configuration space register number 0x4: WRITE_FAILURE - write access to read-only configuration space 0x5: PROTECTED_FAILURE - protected configuration	RO
	3:0	byte_enable	Enable bit per byte bit 0 - byte 0 (bits 7:0) bit 1 - byte 1 (bits 15:8) bit 2 - byte 2 (bits 23:16) bit 3 - byte 3 (bits 31:24) Reserved for read operation.	WO
10h	1:0	ext	Extended or regular configuration space, as defined in PCIE spec: 0 - Regular 1 - Extended	Index
14h	31:0	data		RW

28.29.6 [Internal] MISU – Management Internal Stack Utilization Register

This register is supported by FW for Mellanox usage

MISU Register allows querying for the Firmware stack utilization and is used for firmware verification and debug. Not to be exposed out of Mellanox.

Table 1504 - MISU - Management Internal Stack Utilization Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
						valid																												00h
total_used<0>																total_size<0>																04h		
total_used<1>																total_size<2>																08h		
...																...																...		
total_used<9>																total_size<9>																28h		

Table 1505 - MISU - Management Internal Stack Utilization Register Fields

Offset	Bits	Name	Description	Access
00h	25:16	valid	bitwise vector of which iRISC is valid	RO
	31:16	total_used	Maximum consumed stack (watermark) Units: bytes	RO
	15:0	total_size	Stack size Units: bytes	RO

28.29.7 MPPF - Management PCIe Physical Functions Register

The MPPF register retrieves physical functions (PFs) on a specific port.

The register can return both, the current active PFs on a specified port, or the configured PFs according to the NV_GLOBAL_PCI_CONF TLV (will be active only after reset).

The register supported according to MTCAM register.

Table 1506 - MPPF - Management PCIe Physical Functions Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																				
mode		depth				pcie_index				node				local_port																00h																						
																				num_of_pfs												04h																				
																																																				08h-0Ch
																																								pf[0]								10h				
																																												pf[1]								14h

Table 1507 - MPPF - Management PCIe Physical Functions Register Fields

Offset	Bits	Name	Description	Access
00h	31:30	mode	Specify which PFs configuration to be retrieved. 0x0: CURRENT - current configuration. 0x1: CONFIGURED - next configuration according to NV_GLOBAL_PCI_CONF TLV.	Index
	29:24	depth	depth level of the DUT of some hierarchy	Index
	23:16	pcie_index	PCIe index number (Internal domain index) Reserved when access is from the host	Index
	15:8	node	the node within each depth.	Index
	7:0	local_port	Local port number	Index
04h	8:0	num_of_pfs	Number of PFs in the PFs list.	RO
10h-14h	7:0	pf[2]	Physical functions on the specific local_port. The size of the list reported in num_of_pfs field.	RO

28.30 Pulse Per Second Registers

This section introduces the registers that are used to configure and deal with PPS.

[Internal] For HCA: for more information please refer to Section 13.3.1.1, “Auxiliary Pulse-Per-Second (PPS) Signals,” on page 833.

28.30.1 MTPPS – Management Pulse Per Second Register

This register provides the device PPS capabilities, configure the PPS in and out modules and holds the PPS in time stamp.

[Internal] Spectrum-1 vs. Spectrum-2:

[Internal] PPS out and virtual_PPS are the same HW signal, so no support of NPPS. Spectrum-2 does support.

[Internal] Spectrum-1 supports only free running clock. Spectrum-2 supports also UTC. When UTC only periodic mode.

For HCA: Query of the register supported when HCA_CAP.pps==1. Modify for the register supported when HCA_CAP.pps_modify==1.

Table 1508 - MTPPS - Management PulsePer Second Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
												cap_num- ber_of_pps_ pins								cap_max- num_of_pp s_in_pins								cap_max- num_of_pp s_out_pins				00h	
																																04h	
				cap_pin_3_- mode								cap_pin_2_- mode								cap_pin_1_- mode								cap_pin_0_- mode				08h	
				cap_pin_7_- mode								cap_pin_6_- mode								cap_pin_5_- mode								cap_pin_4_- mode				0Ch	
field_select																																10h	
																																14h-1Ch	
enable												pattern								pin_mode				pin								20h	
																																	24h
	time_stamp																																28h-2Ch
	out_pulse_duration																out_periodic_adjustment																30h
	enhanced_out_periodic_adjustment																																34h

Table 1508 - MTPPS - Management PulsePer Second Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																38h
																																npps

Table 1509 - MTPPS - Management PulsePer Second Register Fields

Offset	Bits	Name	Description	Access
00h	19:16	cap_number_of_pps_pins	Total number of pps pins supported on device. The virtual pins are not counted here	RO
	11:8	cap_max_num_of_pps_in_pins	Maximum number of pps in pins supported on device.	RO
	3:0	cap_max_num_of_pps_out_pins	Maximum number of pps out pins supported on device.	RO
08h	27:24	cap_pin_3_mode	Capability of PPS pin #3 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
	19:16	cap_pin_2_mode	Capability of PPS pin #2 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
	11:8	cap_pin_1_mode	Capability of PPS pin #1 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
	3:0	cap_pin_0_mode	Capability of PPS pin #0 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO

Table 1509 - MTPPS - Management PulsePer Second Register Fields

Offset	Bits	Name	Description	Access
0Ch	27:24	cap_pin_7_mode	Capability of PPS pin #7 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin [Internal] For Switches, only pin 7 is a virtual pin, and it is only virtual pin	RO
	19:16	cap_pin_6_mode	Capability of PPS pin #6 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
	11:8	cap_pin_5_mode	Capability of PPS pin #5 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
	3:0	cap_pin_4_mode	Capability of PPS pin #4 Bit0: support_pps_in Bit1: support_pps_out Bit2: support_virtual_pin	RO
10h	31:0	field_select	Field select indicates which fields to modify in write operation to this register. This field selector supported only when MCAM.mttps_fs==1. if MCAM.mttps_fs==0 then SW should ensure that all the WO/RW fields are valid. [Internal] If field_select==0, then device must look at all fields. Bit 0: enable Bit 1: utc_mode Bit 2: pattern Bit 3: pin_mode Bit 4: time_stamp Bit 5: out_pulse_duration Bit 6: out_periodic_adjustment Bit 7: enhanced_out_periodic_adjustment Bit 8: npps	WO

Table 1509 - MTPPS - Management PulsePer Second Register Fields

Offset	Bits	Name	Description	Access
20h	31	enable	Enables the PPS functionality the specific pin. 0x0: Disable 0x1: Enable	RW
	30	utc_mode	[Internal] For next-gen devices: Connect-EN(?), Spectrum-2 etc. UTC time: 0x0: FREE_RUNNING - time_stamp is by free running clock 0x1: UTC - time_stamp is UTC	RW
	19:16	pattern	When <i>pin_mode</i> = pps_in: 0x0: Rising edge 0x1: Falling edge When <i>pin_mode</i> = pps_out: 0x0: Pulse – (one shot) The device will generate a pulse at <i>time_stamp</i> time with a duration of <i>out_pulse_duration</i> . 0x1: Periodic – the device will generate periodical pulses with pulse duration of <i>out_pulse_duration</i> every 1 second according to internal clock. The 1st pulse will be generated at <i>time_stamp</i> time. Reserved when <i>pin_mode</i> = 0x2 virtual_pin [Internal] When <i>utc_mode</i> = UTC only Periodic is supported (due to HW support) [Internal] Editor note: instead of “every 1 second” need “every <i>pulse_interval</i> ” where <i>pulse_interval</i> is a new field	RW
	11:8	pin_mode	Pin mode to be used. The mode must comply with the supported modes of the requested pin. 0x0: pps_in 0x1: pps_out 0x2: virtual_pin 0x3: pps_out_and_virtual_pin [Internal]: Spectrum-1 supports only virtual_pin and only on pin=7 Operation will fail if pin does not support the required mode.	RW
	7:0	pin	Pin to be configured or queried out of the supported pins.	Index

Table 1509 - MTPPS - Management PulsePer Second Register Fields

Offset	Bits	Name	Description	Access
28h-2Ch	64	time_stamp	When pin_mode = pps_in, the latched device time when it was triggered from the external GPIO pin. When pin_mode = pps_out or virtual_pin or pps_out_and_virtual_pin, the target time to generate next output signal. When utc_mode = FREE_RUNNING - Time is in units of device clock. [Internal] When utc_mode = UTC - Bits 29:0 are nSec, bits 47:30 are Sec [Internal] For HCAs, the device clock frequency can be fetched from HCA_CAP.device_frequency. For Spectrum-1 the device clock frequency is 6.4nSec	RW
30h	31:16	out_pulse_duration	Output pulse duration in units of mSec. Range is 1 to 1000. Reserved when pin_mode != ptp_out	RW
	15:0	out_periodic_adjustment	Frequency adjustment: Every PPS the HW frequency will be adjusted by this value. Units of HW clock, where HW counts 10^9 HW clocks for 1 HW second. Range is from -32768 to +32767 The hardware does consecutive frequency adjustment Reserved when pin_mode != ptp_out Reserved when switches [Internal] Editor note: isn't this a global field for all pin's???	RW
34h	31:0	enhanced_out_periodic_adjustment	Same as out_periodic_adjustment field, but supports a wider range. supported only when MCAM.mttps_enhanced_out_periodic_adjustment==1. mutual exclusive with out_periodic_adjustment, i.e. when enhanced_out_periodic_adjustment!=0, out_periodic_adjustment is invalid. Reserved when switches	RW
38h	7:0	npps	[Internal] For next-gen devices: Connect-EN(?), Spectrum-2 etc.NPPS - N pulse per seconds Range 1: 100 Note: value of 0 is the same as 1	RW

28.30.2 MTPPSE – Management Pulse Per Second Event Register

This register provides configures the different event generation modes for PPS.

For HCA: The register supported when HCA_CAP.pps==1.

Table 1510 - MTPPSE - Management Pulse Per Second Event Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								pin				00h				
event arm																											event_generation_mode		04h			
																													08h-0Ch			

Table 1511 - MTPPSE - Management Pulse Per Second Event Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	pin	Pin to be configured or queried out of the supported pins.	Index
04h	31	event_arm	If set, arm the event mechanism to generate event. Valid only when <i>event_generation_mode</i> = <i>once_till_arm</i> . Note that by default the event is not armed.	RW
	3:0	event_generation_mode	Event generation mode. 0x0: disable - no event is generated.(default mode). 0x1: repetitive_events - generate event on each event regardless event arming 0x2: once_till_arm - generate a single event and wait for rearming	RW

28.30.3 MTUTC – Management UTC Register

This register configures the HW UTC counter.

Table 1512 - MTUTC - Management UTC Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																												operation			00h	
freq_adjustment																												04h				
																												08h-0Ch				
utc_sec																												10h				
																												utc_nsec			14h	
time_adjustment																												18h				

Table 1513 - MTUTC - Management UTC Register Fields

Offset	Bits	Name	Description	Access
00h	3:0	operation	Operation: 0: SET_TIME_AT_NEXT_SEC - the <i>utc_sec</i> will be sampled at the next PPS (of either <i>virtual_pin</i> or <i>hw_utc_nsec</i> wrap around) 1: SET_TIME_IMMEDIATE - reserved when Spectrum-1 2: ADJUST_TIME 3: ADJUST_FREQ_UTC 4: ADJUST_FREQ_FRC - Free Running Clock 5: ADJUST_FREQ_BOTH	OP
04h	31:0	freq_adjustment	Frequency adjustment: Every PPS the HW frequency will be adjusted by this value. Units of HW clock, where HW counts 10^9 HW clocks for 1 HW second. Updated when operation is ADJUST_FREQ [Internal] every pps the HW does $(10^9 + \text{freq\_adjustment})$ nSec. For Spectrum-1 the FW needs to set the new <i>time_stamp</i> to the HW and needs to change the clock of the mirroring ports. For Spectrum-2 the FW sets the nominator and denominator of the free_running	RW
10h	31:0	utc_sec	UTC seconds Updated when operation is SET_TIME_IMMEDIATE or at next PPS when operation is SET_TIME_AT_NEXT_SEC	WO

Table 1513 - MTUTC - Management UTC Register Fields

Offset	Bits	Name	Description	Access
14h	29:0	utc_nsec	UTC nSecs Range 0..(10 ⁹ -1) Updated when operation is SET_TIME_IMMEDIATE Reserved when Spectrum-1	WO
18h	31:0	time_adjustment	Time adjustment Units of nSec Range is from -32768 to +32767 Reserved when Spectrum-1 [Internal] Spc-2: FW will hold the emad until the function is done by HW. FW may have to hold up to 330uSec [Internal] field is 32bit in PRM for future	WO

28.31 Management Component

28.31.1 MCQS - Management Component Query Status

This register allows querying the status of a Firmware component.

For HCA: The support for this register in the HCA is indicated by MCAM register.

Table 1514 - MCQS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
last_index_flag				device_index												component_index																00h
																identifier																04h
																									component_status	component_update_state	08h					
	last_update_state_c hanger_type				last_update_state_c hanger_host_id																				device_type				0Ch			

Table 1515 - MCQS Register Fields

Offset	Bits	Name	Description	Access
00h	31	last_index_flag	When set, this component is the last component (highest <i>component_index</i>).	RO
	27:16	device_index	Device number. For gearboxes, the index represents the gearbox die.	Index
	15:0	component_index	Component Index. Values range from 1 to the last component indicated by <i>last_index_flag</i> .	Index

Table 1515 - MCQS Register Fields

Offset	Bits	Name	Description	Access
04h	15:0	identifier	This field uniquely identifies a specific component type. 0x1: BOOT_IMG 0x2: RUNTIME_IMG - [Internal] Not Implemented 0x3: USER_NVCONFIG - [Internal] Not Implemented 0x4: OEM_NVCONFIG 0x5: MLNX_NVCONFIG 0x6: CS_TOKEN 0x7: DBG_TOKEN 0x8: DEV_INFO - [Internal] Not Implemented 0x9: PACKAGE_INFO - [Internal] Not Implemented 0xA: Gearbox 0xB: CC_ALGO - Congestion Control Algorithm Other values are reserved	RO
08h	8:4	component_status	The status of the component: 0x0: NOT_PRESENT - The component is supported by the device but, currently not present 0x1: PRESENT - This component is present. 0x2: IN_USE - The component is present and currently in use. Other values are reserved	RO
	3:0	component_update_state	Component statue in update flow, see “Component Update State” on page 635: 0x0: IDLE 0x1: IN_PROGRESS 0x2: APPLIED 0x3: ACTIVE 0x4: ACTIVE_PENDING_RESET 0x5: FAILED 0x6: CANCELED 0x7: BUSY Other values are reserved	RO

Table 1515 - MCQS Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:28	last_update_state_changer_type	Indicates which entity was the last to change the <i>component_update_state</i> of this component. 0x0: unspecified 0x1: Chassis_BMC 0x2: MAD 0x3: BMC 0x4: command_interface 0x5: ICMD Other values are reserved	RO
	27:24	last_update_state_changer_host_id	For <i>last_update_state_changer_type</i> BMC, command-interface and ICMD, indicates the identifier of the host of the handle owner. Otherwise reserved.	RO
	7:0	device_type	Peripheral device type: 0: Switch_or_NIC 1: Gearbox	Index

28.31.2 MCQI - Management Component Query Information

This register allows querying information about Firmware components.

For HCA: The support for this register in the HCA is indicated by MCAM register.

Table 1516 - MCQI Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
read_pending_component				device_index												component_index																00h	
																									device_type				04h				
																													info_type		08h		
	info_size																																0Ch
	offset																																10h
																	data_size																14h
	data[0]																																18h
	data[1]																																1Ch
	...																																...

Table 1517 - MCQI Register Fields

Offset	Bits	Name	Description	Access
00h	31	read_pending_component	When set, the register will return information about the pending component (if available)	Index
	27:16	device_index	Device number. For gearboxes, the index represents the gearbox die.	Index
	15:0	component_index	Component index gathered by “ MCQS - Management Component Query Status ”	Index
04h	7:0	device_type	Peripheral device type: 0: Switch / NIC 1: Gearbox	Index

Table 1517 - MCQI Register Fields

Offset	Bits	Name	Description	Access
08h	4:0	info_type	Component properties set. 0x0: CAPABILITIES 0x1: VERSION [Internal] 0x2: PUBLIC_KEYS -- structure defined in [SEC_FW] [Internal] 0x3: FORBIDDEN_VERSIONS -- structure defined in [SEC_FW] 0x5: ACTIVATION_METHOD Other values are reserved	RW
0Ch	31:0	info_size	Properties set structure size, given in bytes.	RO
10h	31:0	offset	The requested/returned data offset from the section start, given in bytes. Must be DWORD aligned.	RW
14h	15:0	data_size	The requested/returned data size, given in bytes. If <i>data_size</i> is not DWORD aligned, the last bytes are zero padded.	RW
18h-...	31:0	data[...]	Properties set structure according to <i>info_type</i> . CAPABILITIES - See Table 1518, “MCQI CAPABILITIES Info Layout,” on page 1623 VERSION - See Table 1520, “MCQI VERSION Info Layout,” on page 1625 ACTIVATION_METHOD - See Table 1524, “MCQI ACTIVATION_METHOD Info Layout,” on page 1627	RO

Table 1518 - MCQI CAPABILITIES Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
supported_info_bitmask																															00h	
component_size																															04h	
max_component_size																															08h	
log_mc-da_word_size																mcda_max_write_size											0Ch					

Table 1518 - MCQI CAPABILITIES Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rd_en	signed_updates_only	match_chip_id	match_psid	check_user_timestamp	match_base_guid_mac																											10h

Table 1519 - MCQI CAPABILITIES Info Fields

Offset	Bits	Name	Description	Access
00h	31:0	supported_info_bitmask	Bitmask indicating which info_type(s) are supported for this component. Set bit indicates the property set is supported bit 1: VERSION [Internal] bit 2: PUBLIC_KEYS [Internal] bit 3: FORBIDDEN_VERSIONS bit 5: ACTIVATION_METHOD Other bits are reserved. CAPABILITIES set is always supported.	RO
04h	31:0	component_size	The size of the component given in bytes. Value 0x0 indicates that the size is unknown. For some components, size may only be available in the READ_COMPONENT state.	RO
08h	31:0	max_component_size	Maximum size for this component, given in bytes.	RO
0Ch	31:28	log_mcda_word_size	Log 2 of the access word size in bytes. Read and write access must be aligned to the word size. Write access must be done for an integer number of words.	RO
	15:0	mcda_max_write_size	Maximal write size for MCDA	RO

Table 1519 - MCQI CAPABILITIES Info Fields

Offset	Bits	Name	Description	Access
10h	31	rd_en	When set, this components may be read, see “Read Flow” on page 638 .	RO
	30	signed_updates_only	Only signed components are accepted.	
	29	match_chip_id	Chip ID (<i>device_hw_revision</i>) is validated for this component update.	
	28	match_psid	PSID is validated for this component update.	
	27	check_user_timestamp	A user defined timestamp (MVTs) is active for this component.	
	26	match_base_guid_mac	The device only accepts updates for this component that explicitly lists its base MAC and/or base GUID	

Table 1520 - MCQI VERSION Info Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
			user_defined_time_valid																													00h
		build_time_valid																														04h
																																08h-0Ch
																																10h-14h
																																18h
																																1Ch
																																20h
																																...
																																78h

Table 1521 - MCQI VERSION Info Fields

Offset	Bits	Name	Description	Access
00h	29	build_time_valid	When set, the component has a valid creation time-stamp in <i>build_time</i>	
00h	28	user_defined_time_valid	When set, the component has a valid user-defined version time-stamp in <i>user_defined_time</i>	
00h	7:0	version_string_length		
04h	31:0	version	Version number	
08h-0Ch	64	build_time	Time of component creation. Valid only if <i>build_time_valid</i> is set. See Table 1522, “Date-Time Layout,” on page 1626	
10h-14h	64	user_defined_time	User-defined time assigned to the component version. Valid only if <i>user_defined_time_valid</i> is set. See Table 1522, “Date-Time Layout,” on page 1626	
18h	31:0	build_tool_version	Build tool version number. Valid only when not equal to 0	
20h-78h	31:24	version_string[92]	NULL terminated string representing the version.	

Table 1522 - Date-Time Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset										
seconds								minutes								hours																		00h								
year																month								day																		04h

Table 1523 - Date-Time Fields

Offset	Bits	Name	Description	Access
00h	31:24	seconds	seconds - 2 packed BCD digits	
	23:16	minutes	minutes - 2 packed BCD digits	
	15:8	hours	hour (UTC) - 2 packed BCD digits	

Table 1523 - Date-Time Fields

Offset	Bits	Name	Description	Access
04h	31:16	year	year - 4 packed BCD digits	
	15:8	month	month - 2 packed BCD digits	
	7:0	day	day - 2 packed BCD digits	

Table 1524 - MCQI ACTIVATION_METHOD Info Layout

Offset	00h
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	device hw reset
26	all hosts sync
27	auto activate
28	pending fw reset
29	pending_server reboot
30	pending_server dc power cycle
31	pending_server ac power cycle

Table 1525 - MCQI ACTIVATION_METHOD Info Fields

Offset	Bits	Name	Description	Access
00h	31	pending_ - server_ac_pow- er_cycle	This component goes to ACTIVE_PENDING_RESET state after activation. AC power cycle (power cycle for both PCI power rails and AUX power), will move it ACTIVE state.	RO
	30	pending_ - server_dc_pow- er_cycle	This component goes to ACTIVE_PENDING_RESET state after activation. DC power cycle (power cycle of PCI power rails), will move it ACTIVE state.	RO
	29	pending_ - server_reboot	This component goes to ACTIVE_PENDING_RESET state after activation. A server reset (PCIe PERST#), will move it ACTIVE state.	RO
	28	pending_fw_re- set	This component goes to ACTIVE_PENDING_RESET after activation. A FW reset will move it to ACTIVE state.	RO
	27	auto_activate	This component will be ACTIVE or ACTIVE_PENDING_RESET after the APPLY state.	RO
	26	all_hosts_sync	When set, the operation needed to move the component form ACTIVE_PENDING_RESET to ACTIVE should happen simultaneously on all hosts.	RO
	25	device_hw_reset	This component goes to ACTIVE_PENDING_RESET after activation. A HW reset will move it to ACTIVE state.	RO

28.31.3 MCC - Management Component Control

This register allows controlling the Firmware component update FSM.

For HCA: The support for this register in the HCA is indicated by MCAM register.

Table 1526 - MCC Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
				time_elapsed_since_last_cmd																				instruction								00h	
																component_index												04h					
								update_handle																			08h						
handle_owner_type				handle_owner_host_id					control_progress								error_code											control_state				0Ch	
component_size																																10h	
					device_index																				device_type								14h
																																18h-1Ch	

Table 1527 - MCC Register Fields

Offset	Bits	Name	Description	Access
00h	27:16	time_elapsed_since_last_cmd	The number of seconds elapsed since the update_handle owner last issued a command. The time saturates at 0xFF. [Internal] .INI can specify an “update_handle_timeout_sec” value after-which the handle will expire automatically. The default value for this parameter is 600 seconds (10 minutes), zero means no timeout.	RO
	7:0	instruction	Command to be executed by the FSM 0x1: LOCK_UPDATE_HANDLE 0x2: RELEASE_UPDATE_HANDLE 0x3: UPDATE_COMPONENT 0x4: VERIFY_COMPONENT 0x5: ACTIVATE_COMPONENT 0x6: ACTIVATE 0x7: READ_COMPONENT 0x8: CANCEL 0x9: CHECK_UPDATE_HANDLE 0xA: FORCE_HANDLE_RELEASE 0xB: READ_PENDING_COMPONENT Other values are reserved. Applicable for write operation only.	RW
04h	15:0	component_index	Index of the accessed component. Valid for UPDATE_COMPONENT, ACTIVATE_COMPONENT, READ_COMPONENT and READ_PENDING_COMPONENT instructions. Otherwise, this field is reserved.	Index
08h	23:0	update_handle	Token representing the current flow executed by the FSM. See “Update Handle” on page 636.	RW

Table 1527 - MCC Register Fields

Offset	Bits	Name	Description	Access
0Ch	31:28	handle_owner_type	Type of entity holding the update handle: 0x0: unspecified 0x1: Chassis BMC 0x2: MAD 0x3: BMC 0x4: command interface 0x5: ICMD Other values are reserved.	RO
	27:24	handle_owner_host_id	For <i>handle_owner_type</i> BMC, command-interface and ICMD, indicates the identifier of the host of the handle owner. Otherwise reserved	RO
	22:16	control_progress	Indicates the estimated progress status of the current operation executed by the FSM. Valid values are 0..100. 101 indicates that progress reporting is not supported for this update state.	RO
	15:8	error_code	Indicates the successful completion of the <i>instruction</i> , or the reason it failed. See “Error Handling” on page 638 0x0: OK 0x1: ERROR 0x2: REJECTED_DIGEST_ERR 0x3: REJECTED_NOT_APPLICABLE 0x4: REJECTED_UNKNOWN_KEY 0x5: REJECTED_AUTH_FAILED 0x6: REJECTED_UNSIGNED 0x7: REJECTED_KEY_NOT_APPLICABLE 0x8: REJECTED_BAD_FORMAT 0x9: BLOCKED_PENDING_RESET 0xA: REJECTED_NOT_A_SECURED_FW 0xB: REJECTED_MFG_BASE_MAC_NOT_LISTED 0xC: REJECTED_NO_DEBUG_TOKEN 0xD: REJECTED_VERSION_NUM_MISMATCH 0xE: REJECTED_USER_TIMESTAMP_MISMATCH 0xF: REJECTED_FORBIDDEN_VERSION 0x10: FLASH_ERASE_ERROR 0x11: REJECTED_REBURN_RUNNING_AND_RETRY Other values should be treated as an unknown error.	RO

Table 1527 - MCC Register Fields

Offset	Bits	Name	Description	Access
0Ch	3:0	control_state	Current Update FSM state, See “FSM States” on page 639 0x0: IDLE 0x1: LOCKED 0x2: INITIALIZE 0x3: DOWNLOAD 0x4: VERIFY 0x5: APPLY 0x6: ACTIVATE 0x7: UPLOAD 0x8: UPLOAD_PENDING Other values are reserved	RO
10h	31:0	component_size	Component size in bytes. Valid for UPDATE_COMPONENT instruction. Specifying the size may shorten the update time. Value 0x0 means that size is unspecified.	WO
14h	27:16	device_index	Device number. For gearboxes, the index represents the gearbox die.	Index
	7:0	device_type	Peripheral device type: 0: Switch / NIC 1: Gearbox	Index

28.31.4 MCDA - Management Component Data Access

This register allows reading and writing a Firmware Component.

For HCA: The support for this register in the HCA is indicated by MCAM register.

Table 1528 - MCDA Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h
																																...
																																8Ch

Table 1529 - MCDA Register Fields

Offset	Bits	Name	Description	Access
00h	23:0	update_handle	Update handle registered when the FSM was activated.	RW
04h	31:0	offset	Offset of accessed address relative to component start. Accesses must be in accordance to <i>log_mcd_a_word_size</i> in Table 1518, “MCQI CAPABILITIES Info Layout,” on page 1623	RW
08h	15:0	size	Size of the data accessed, given in bytes	RW
10h-8Ch	31:0	data[32]	Data block accessed	RW

28.31.5 MQIS - Management Query Information Strings

This register allows querying general information about the device, given in a string format.

For HCA: The support for this register in the HCA is indicated by **MCAM register**.

Table 1530 - MQIS Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								info_type				00h				
																info_length												04h				
read_offset																read_length												08h				
																																0Ch
info_string[0]								info_string[1]								info_string[2]								info_string[3]								10h
info_string[4]								info_string[5]								info_string[6]								info_string[7]								14h

Table 1531 - MQIS Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	info_type	Type of information string to be queried: 0x0: VPD - Read the PCI Vital Product Data capability content. 0x1: MODEL_NAME 0x2: MODEL_DESCRIPTION 0x3: IMAGE_VSD 0x4: DEVICE_VSD 0x5: ROM_INFO other values are reserved.	Index
04h	15:0	info_length	Total size of the information string, according to <i>info_type</i> . Value given in bytes.	RO
08h	31:16	read_offset	Offset in bytes of the first byte requested to read.	Index
	15:0	read_length	Number of bytes requested. The device returns the number of bytes actually read.	RW
10h-14h	31:24	info_string[8]	Information string accessed, according to <i>info_type</i> . If the information is not available, a NULL string is returned.	RO

28.31.6 MIRC - Management Image Re-activation Control Register

This register allows controlling the fail-safe method flow to re-activate a running FW and allow to program a new FW version instead of an already activated FW which is not yet in use.

Table 1532 - MIRC Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h

Table 1533 - MIRC Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	status_code	Indicates the successful completion of the instruction, or the reason it failed. See “Error Handling” on page 638 0x0: OK 0x1: BUSY 0x2: PROHIBITED_FW_VER_ERR 0x3: FIRST_PAGE_COPY_FAILED 0x4: FIRST_PAGE_ERASE_FAILED 0x5: FIRST_PAGE_RESTORE_FAILED 0x6: CANDIDATE_FW_DEACTIVATION_FAILED 0x7: FW_ALREADY_ACTIVATED 0x8: ERROR_DEVICE_RESET_REQUIRED 0x9: ERROR_FW_PROGRAMMING_NEEDED Other values should be treated as an unknown error.	RO

28.31.7 MCDD - Management Component DMA Based Data Access

The register used to initiate programming of FW component data section which was pre-loaded into the host memory by the host software.

Table 1534 - MCDD - Management Component DMA Based Data Access Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h
																																0Ch
																																10h
																																14h
																																18h
																																1Ch

Table 1535 - MCDD - Management Component DMA Based Data Access Register Fields

Offset	Bits	Name	Description	Access
00h	23:0	update_handle	Update handle registered when the FSM was activated.	RW
04h	31:0	offset	Offset of accessed address relative to component start.	RW
08h	15:0	size	Size of the data accessed, given in bytes	RW
10h	31:0	data_page_- phys_ad- dr_msb	Physical address of the FW component data page.	RW
14h	31:0	data_page_- phys_addr_lsb		RW
18h	31:0	mailbox- _page_- phys_addr_ms b	Physical address of the mailbox page.	RW
1Ch	31:0	mailbox- _page_- phys_addr_lsb		RW

28.31.8 MDFCR - Management Peripheral Devices Firmware Control Register

MDFCR register allows software to query and control the burning flow of the firmware for peripheral devices as gearboxes, retimers, COBO etc.

Table 1536 - MDFCR - Management Peripheral Devices Firmware Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								device_type				00h				
all																					device_index						04h					
																																08
								fw_status												first_fw_status_device								0Ch				
expected_fw_version																																10h
fw_version																																14h
																build_id																18h
																								major				1Ch				
																								minor				20h				
																error_id												24h				
reserved																																28h-5Ch

Table 1537 - MDFCR - Management Peripheral Devices Firmware Control Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	device_type	Peripheral device type 1: Vega_Gearbox 2: Tile	index
04h	31	all	All devices are selected and <i>device_index</i> is ignored When set to '1', the <i>fw_status</i> should represent the worst case of any of the devices. <i>fw_version</i> , <i>build_id</i> , <i>major</i> and <i>minor</i> should be of device number 0.	index
	11:0	device_index	Device number For gearboxes, the index represents the gearbox die.	index

Table 1537 - MDFCR - Management Peripheral Devices Firmware Control Register Fields

Offset	Bits	Name	Description	Access
0Ch	23:20	fw_status	FW load status. 0: No errors, FW has loaded successfully. [Internal]: for Vega gearbox in 0xACC0 1: Flash error. [Internal]: For Vega gearbox, after two recoveries still in 0xBCDD status. Vega HW flash is malfunctioning. 2: FW load error. [Internal]: For Vega gearbox, after 5 sec still in 0xBCEE 15: Not ready yet for displaying the status.	RO
	11:0	first_fw_status_device	The first device representing the <i>fw_status</i>	RO
10h	31:0	expected_fw_version	Expected FW version	RO
14h	31:0	fw_version	FW version	RO
18h	15:0	build_id	FW build ID	RO
1Ch	7:0	major	FW major version	RO
20h	7:0	minor	FW minor version	RO
24h	15:11	error_id	FW burning status error. [Internal]: For Vega gearbox these errors are aligned to the data-sheet. For other projects, ENUMs are not aligned and should be converted accordingly. 1: no_errors 2: error_in_total_size 3: error_in_iRAM_size_read 4: error_in_DRAM_size_read 5: SPI_read_from_EEPROM_time-out 6: CRC_on_downloaded_EEPROM_content 7: Download_process_is_on-going 8: SPI_reset_time-out 9: memory_protection 10: unknown_parameter_was_received_by_bootloader 11: invalid_message_was_received_by_bootloader 12: message_was_aborted_by_host Values from 13-32 are reserved	RO

28.31.9 MDCR - Management Peripheral Devices Control Register

MDCR register allows software to verify and control the peripheral devices.

Table 1538 - MDCR - Management Devices Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								device_type				00h				
					op															device_index				04h								
																																08h-0Ch
device_status_data(see Table 1540, “MDCR - XM Data Layout,” on page 1639)																																10h-5Ch

Table 1539 - MDCR - Management Devices Control Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	device_type	Peripheral device type 1: XM	index
04h	27:24	op	Special op-code. When used, <i>device_status_data</i> field might be irrelevant. 1: Ignore mismatch of FW version from expected and proceed.	OP
04h	11:0	device_index	Device number For gearboxes, the index represents the gearbox die.	index
10h-5Ch	640	device_status_data(see Table 1540, “MDCR - XM Data Layout,” on page 1639)	Device initialization Data. For XM status, see Table 1540, “MDCR - XM Data Layout,” on page 1639	RO

Table 1540 - MDCR - XM Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								xm_status																								00h
																xm_major														04h		
																xm_minor														08		

Table 1540 - MDCR - XM Data Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																xm_build_id																0Ch
																xm_expected_major																10h
																xm_expected_minor																14h
reserved																																18h-4Ch

Table 1541 - MDCR - XM Data Fields

Offset	Bits	Name	Description	Access
00h	23:20	xm_status	XM load status. 0: Pending 1: No connection (At least 1 of the two Ethernet ports is not up) 2: Version mismatch (<i>major</i> and <i>minor</i> are different from <i>major_expected</i> and <i>minor_expected</i>)	RO
04h	15:0	xm_major	XM major version	RO
08h	15:0	xm_minor	XM minor version	RO
0Ch	15:0	xm_build_id	XM build ID	RO
10h	15:0	xm_expected_major	XM expected major version	RO
14h	15:0	xm_expected_minor	XM expected minor version	RO

28.31.10MDRCR - Management Peripheral Devices Reset Control Register

MDRCR register allows software to initialize reset and query the status of it for peripheral devices as gearboxes, retimers, COBO etc.

Table 1542 - MDRCR - Management Peripheral Devices Reset Control Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								device_type				00h				
all																					device_index						04h					
																																08h
												rst_status														rst_op		0Ch				
reserved																																10h-2Ch

Table 1543 - MDRCR - Management Peripheral Devices Reset Control Register Fields

Offset	Bits	Name	Description	Access
00h	7:0	device_type	Peripheral device type 1: Gearbox [internal]: 2: Tile	index
04h	31	all	All devices are selected and <i>device_index</i> is ignored When set to '1', the <i>rst_status</i> should represent the worst case of any of the devices.	index
	11:0	device_index	Device number For gearboxes, the index represents the gearbox die.	index
0Ch	19:16	rst_status	Reset status. 0: Reset completed successfully. [Internal]: For Vega gearboxes, once reached checking the FW, this status should be set. 1: Reset operation has not finished yet.	RO
	2:0	rst_op	Reset Operation. 0: Query of the reset status which is displayed through <i>rst_status</i> . 1: Full Reset of the selected device(s). 2-8: Reserved.	RW

28.31.11MDSR - Management Debug Session Register

This register is used to query the debug session status .

Table 1544 - MDSR - Management Debug Session Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-2Ch

Table 1545 - MDSR - Management Debug Session Register Fields

Offset	Bits	Name	Description	Access
00h	13:8	addition- al_info	0: No additional information available 1: There is no debug session in progress 2: FW is not secured, debug session cannot be ended 3: Fail - Request was sent from a different host that the one that issued the debug session 4: Fail - Host is not allowed to query debug session	RO
	3:0	status	0: The debug session ended successfully 1: Failed to execute the operation. See <i>additional_info</i> for more details. 2-15: Reserved	RO
04h	31	end	Set to '1' to end debug session.	WO

28.32 General Management

28.32.1 MCAM - Management Capabilities Mask Register

The MCAM register reports the device supported management features.

For HCA: This register is supported only when `HCA_CAP.MCAM == 1`.

Table 1546 - MCAM - Management Capabilities Mask Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h
																																08h-14h
																																18h-24h
																																28h-34h
																																38h-44h

Table 1547 - MCAM - Management Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
00h	23:16	feature_group	Feature list mask index: 0: enhanced_features	Index
	7:0	access_reg_group	Access Register ID groups 0: First_128_REG_ID - Register IDs 0x9001 - 0x907F) 1: Register_IDs_0x9080 - 0x90FF (bit 0 in <i>mng_access_reg_cap_mask</i> represent register ID 0x9080 while bit 127 represents register ID 0x90FF). 2: Register_IDs_0x9100 - 0x917F (bit 0 in <i>mng_access_reg_cap_mask</i> represent register ID 0x9100 while bit 127 represents register ID 0x917F).	Index

Table 1547 - MCAM - Management Capabilities Mask Register Fields

Offset	Bits	Name	Description	Access
08h-14h	128	mng_access_reg_cap_mask	Supported management's access register bitmasks. Based on <i>access_reg_group</i> index. When bit is set, The register is supported in the device. For example: Bit 1: MFCR_0x9001 Bit 2: MFSC_0x9002 Bit 3: MFSM_0x9003 Bit 4: MFSL_0x9004 Bit 58: MGCR_0x903A Bit 73: MPPF_0x9049 Bit 127: MCAP_0x907F	RO
28h-34h	128	mng_feature_cap_mask	Supported port's enhanced features. Based on <i>feature_group</i> index. When bit is set, The feature is supported in the device: Bit 0: MPCNT counter group- PCIE performance counters supported Bit 1: mttps_fs - If set, field_select field in MTPPS register is supported. Bit 2: mttps_enhanced_out_periodic_adjustment - If set, enhanced_out_periodic_adjustment field in MTPPS register is supported. Bit 3: tx_lossy_overflow_oper - If set, tx_overflow_buffer_pkt counter in MPCNT register is supported. Bit 4: pcie_outbound_stalled - if set, outbound_stalled_reads, outbound_stalled_writes, outbound_stalled_reads_events and outbound_stalled_writes_events counters in MPCNT are supported. Bit 5: Management pass through is supported Bit 6: sensor_map - If set, sensor_map is supported in MTCAP register. Bit 7 - if set, module_status bit 8 (Module Low Power) in MCION register is supported. Bit 8 - beacon_capability_disable - If set, beacon feature, as appears in MLCR register, is not supported by the device. Bit 9 - dynamic_tx_overflow - If set, tx_overflow_sense field is supported in MPEGC register. Bit 10 - mark_tx_action_cqe is supported if set to '1'. Bit 11 - mark_tx_action_cnp is supported if set to '1'. Bit 12 - dev_info is supported in register is set to '1'. Bit 13 - sensor_count field is 12bit size in MTMP and MTBR Bit 14 - cs_tokens_supported is supported Bit 15 - debug_fw_tokens_supported is supported Bit 16 - long_keys is supported Bit 17 - pwr_status and pci_power are supported in MPEIN Bit 18 - If set, accessing through device_type and device_index is supported in MCC, MCQI and MCQS	RO

28.32.2 MFRL - Management Firmware Reset Level

Image update is combined from two stages, flashing of the new image and activating (e.g. starting) the use of the new image.

The MFRL register allows the user to instruct the NIC to activate a new image.

If update of non-volatile configuration is needed, MFRL can be used to reset the device and re-acquire non-volatile configuration.

The device reset supports multiple levels, where the lower level provides a softer reset (less intrusive) than a high-level reset.

The minimum required reset-level is reported by the device according to the difference between current image and the new, pending, image. A user might use the device recommended reset-level or use a higher-level reset.

The device also offers a mechanism to sync all functions on a device reset. This mechanism allows graceful handling in the device driver

Table 1548 - MFRL - Management Firmware Reset Level Register Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
																																00h			
pci_sync_for_fw_update_start		pci_sync_for_fw_update_resp		rst_type_sel												reset_type										reset_level									
																										04h									

Table 1549 - MFRL - Management Firmware Reset Level Register Fields

Offset	Bits	Name	Description	Access
04h	29	pci_sync_for_fw_update_start	Setting this bit to '1' will request a synced driver flow and '0' for the legacy flow	WO
	28:27	pci_sync_for_fw_update_resp	Response of the driver for the reset request. 1: Acknowledgment 2: Dis-acknowledgment 3: Reserved	WO
	26:24	rst_type_sel	The requested reset.	RW
	15:8	reset_type	Each bit represents a chip reset type. If set to '1', the reset is supported. Bit 0: Full chip reset Bit 1: Port alive	RO
	7:0	reset_level	The firmware reset level. See “MFRL - Management Firmware Reset Level” on page 1645 Only a single bit may be set. Bit 0: LEVEL0 Bit 3: LEVEL3 Bit 6: LEVEL6 Other bits are reserved.	RW

29 Command Reference

This chapter is a reference for all the configuration commands that can be executed via the device's command interface. For each command, the chapter describes the input and output arguments and certain restrictions (such as whether an event can be generated for the command completion).

These commands are used to read HCA capabilities and status and to configure the HCA. This is in addition to MAD interface also used for those purposes.

29.1 Introduction

Notes:

- Reserved fields on input must be set to 0h
- Reserved fields on output must be ignored by software
- Fields that appear void are considered reserved for every practical purpose
- When a command is not completed successfully, output data is undefined, unless otherwise specified

Table 1550 lists all the commands described in this chapter sorted by the opcode along with links to the respective sections describing the commands. [Note - 0x900-0x9ff saved for Shomron.](#)

Table 1550 - Commands List Sorted by Opcode (Sheet 1 of 11)

Group	Opcode	Command Name	Link to Command Description
Init	0x100	QUERY_HCA_CAP	Section 29.3.4, on page 1667
Init	0x101	QUERY_ADAPTER	Section 29.3.12, on page 1816
Init	0x102	INIT_HCA	Section 29.3.13, on page 1820
Init	0x103	TEARDOWN_HCA	Section 29.3.14, on page 1821
Init	0x104	ENABLE_HCA	Section 29.3.15, on page 1823
Init	0x105	DISABLE_HCA	Section 29.3.16, on page 1825
Init	0x107	QUERY_PAGES	Section 29.3.1, on page 1660
Init	0x108	MANAGE_PAGES	Section 29.3.2, on page 1662
Init	0x109	SET_HCA_CAP	Section 29.3.5, on page 1803
Init	0x10A	QUERY_ISSI	Section 29.3.17, on page 1826
Init	0x10B	SET_ISSI	Section 29.3.18, on page 1828
Init	0x10C	QUERY_DRIVER_VERSION	Section 29.3.20, on page 1830
Init	0x10D	SET_DRIVER_VERSION	Section 29.3.19, on page 1829
Init	0x10E	QUERY_OTHER_HCA_CAP	Section 29.3.6, on page 1807

Table 1550 - Commands List Sorted by Opcode (Sheet 2 of 11)

Group	Opcode	Command Name	Link to Command Description
Init	0x10F	MODIFY_OTHER_HCA_CAP	Section 29.3.7, on page 1809
Init	0x110	SET_TUNNELED_OPERATIONS	
Init	0x111	QUERY_SF_PARTITIONS	Section 29.3.8, on page 1810
Init	0x112	SET_SF_PARTITIONS	Section 29.3.9, on page 1812
Init	0x113	ALLOC_SF	Section 29.3.10, on page 1814
Init	0x114	DEALLOC_SF	Section 29.3.11, on page 1815
TPT	0x200	CREATE_MKEY	Section 29.6.1, on page 1861
TPT	0x201	QUERY_MKEY	Section 29.6.2, on page 1865
TPT	0x202	DESTROY_MKEY	Section 29.6.3, on page 1868
TPT	0x203	QUERY_SPECIAL_CONTEXTS	Section 29.6.5, on page 1871
TPT	0x204	PAGE_FAULT_RESUME	Section 29.6.4, on page 1869
MEMIC	0x205	ALLOC_MEMIC	Section 29.6.6, on page 1873
MEMIC	0x206	DEALLOC_MEMIC	Section 29.6.7, on page 1874
EQ	0x301	CREATE_EQ	Section 29.8.1, on page 1880
EQ	0x302	DESTROY_EQ	Section 29.8.2, on page 1883
EQ	0x303	QUERY_EQ	Section 29.8.3, on page 1884
EQ	0x304	GEN_EQE	Section 29.8.4, on page 1886
CQ	0x400	CREATE_CQ	Section 29.9.1, on page 1888
CQ	0x401	DESTROY_CQ	Section 29.9.2, on page 1890
CQ	0x402	QUERY_CQ	Section 29.9.3, on page 1891
CQ	0x403	MODIFY_CQ	Section 29.9.4, on page 1894
QP	0x500	CREATE_QP	Section 29.10.1, on page 1899
QP	0x501	DESTROY_QP	Section 29.10.2, on page 1904
QP	0x502	RST2INIT_QP	Section 29.10.3, on page 1905
QP	0x503	INIT2RTR_QP	Section 29.10.5, on page 1909
QP	0x504	RTR2RTS_QP	Section 29.10.6, on page 1914
QP	0x505	RTS2RTS_QP	Section 29.10.7, on page 1917
QP	0x506	SQERR2RTS_QP	Section 29.10.8, on page 1922
QP	0x507	QP_2ERR	Section 29.10.9, on page 1924

Table 1550 - Commands List Sorted by Opcode (Sheet 3 of 11)

Group	Opcode	Command Name	Link to Command Description
QP	0x50A	QP_2RST	Section 29.10.10, on page 1925
QP	0x50B	QUERY_QP	Section 29.10.11, on page 1927
QP	0x50C	SQD2RTS_QP	Section 29.10.12, on page 1929
QP	0x50E	INIT2INIT_QP	Section 29.10.4, on page 1907
QP	0x50F	SUSPEND_QP	Section 29.10.13, on page 1931
QP	0x510	RESUME_QP	Section 29.10.14, on page 1932
QP	0x511	SET_RDB	Section 29.10.15, on page 1933
QP	0x512	QUERY_RDB	Section 29.10.16, on page 1935
QP	0x513	MUST_NOT_USE	
QP	0x514	MUST_NOT_USE2	
SIG	0x600	CREATE_PSV	Section 29.7.1, on page 1876
SIG	0x601	DESTROY_PSV	Section 29.7.2, on page 1878
SIG	0x602	QUERY_PSV	Section 29.7.3, on page 1879
SRQ	0x700	CREATE_SRQ	Section 29.25.1, on page 2103
SRQ	0x701	DESTROY_SRQ	Section 29.25.2, on page 2105
SRQ	0x702	QUERY_SRQ	Section 29.25.3, on page 2106
SRQ	0x703	ARM_RQ	Section 29.25.4, on page 2108
SRQ	0x704	RESIZE_SRQ	Section 22.19.4, on page 707
XRC_SRQ	0x705	CREATE_XRC_SRQ	Section 29.26.1, on page 2110
XRC_SRQ	0x706	DESTROY_XRC_SRQ	Section 29.26.2, on page 2112
XRC_SRQ	0x707	QUERY_XRC_SRQ	Section 29.26.3, on page 2113
XRC_SRQ	0x708	ARM_XRC_SRQ	Section 29.26.4, on page 2115
DCT	0x709	CREATE_DCT_LAG	Section 29.30.1, on page 2144
DCT	0x710	CREATE_DCT	Section 29.30.1, on page 2144
DCT	0x711	DESTROY_DCT	Section 29.30.2, on page 2145
DCT	0x712	DRAIN_DCT	Section 29.30.4, on page 2147
DCT	0x713	QUERY_DCT	Section 29.30.3, on page 2146
DCT	0x714	ARM_DCT_FOR_KEY_VIOLA-TION	Section 29.30.5, on page 2148

Table 1550 - Commands List Sorted by Opcode (Sheet 4 of 11)

Group	Opcode	Command Name	Link to Command Description
DCT	0x715	SET_DC_CNAK_TRACE	Section 29.30.7, on page 2151
DCT	0x716	QUERY_DC_CNAK_TRACE	Section 29.30.8, on page 2153
XRQ	0x717	CREATE_XRQ	Section 29.27.1, on page 2117
XRQ	0x718	DESTROY_XRQ	Section 29.27.2, on page 2118
XRQ	0x719	QUERY_XRQ	Section 29.27.3, on page 2119
NVMe-oF	0x720	CREATE_NVMF_BACKEND_CONTROLLER	Section 29.28.1, on page 2129
NVMe-oF	0x721	DESTROY_NVMF_BACKEND_CONTROLLER	Section 29.28.2, on page 2132
NVMe-oF	0x722	QUERY_NVMF_BACKEND_CONTROLLER	Section 29.28.2, on page 2132
NVMe-oF	0x723	ATTACH_NVMF_NAMESPACE	Section 29.28.4, on page 2134
NVMe-oF	0x724	DETACH_NVMF_NAMESPACE	Section 29.28.5, on page 2136
XRQ	0x725	QUERY_XRQ_DC_PARAMS_ENTRY	Section 29.27.5, on page 2122
XRQ	0x726	SET_XRQ_DC_PARAMS_ENTRY	Section 29.27.6, on page 2124
XRQ	0x727	QUERY_XRQ_ERROR_PARAMS	Section 29.27.7, on page 2125
NVMe-oF	0x728	QUERY_NVMF_NAMESPACE	Section 29.28.6, on page 2137
XRQ	0x729	RELEASE_XRQ_ERROR	Section 29.27.8, on page 2128
XRQ	0x72A	MODIFY_XRQ	Section 29.27.8, on page 2128
VPORT	0x740	QUERY_ESW_FUNCTIONS	Section 29.29, on page 2140
VPORT	0x750	QUERY_VPORT_STATE	Section 29.31.1, on page 2155
VPORT	0x751	MODIFY_VPORT_STATE	Section 29.31.2, on page 2157
VPORT	0x752	QUERY_ESW_VPORT_CONTEXT	Section 29.31.3, on page 2158
VPORT	0x753	MODIFY_ESW_VPORT_CONTEXT	Section 29.31.4, on page 2160
VPORT	0x754	QUERY_NIC_VPORT_CONTEXT	Section 29.31.5, on page 2162
VPORT	0x755	MODIFY_NIC_VPORT_CONTEXT	Section 29.31.6, on page 2164
RoCE	0x760	QUERY_ROCE_ADDRESS	Section 29.27.5, on page 2122

Table 1550 - Commands List Sorted by Opcode (Sheet 5 of 11)

Group	Opcode	Command Name	Link to Command Description
RoCE	0x761	SET_ROCE_ADDRESS	Section 29.33, on page 2175
VPORT	0x762	QUERY_HCA_VPORT_CONTEXT	Section 29.31.7, on page 2166
VPORT	0x763	MODIFY_HCA_VPORT_CONTEXT	Section 29.31.8, on page 2167
VPORT	0x764	QUERY_HCA_VPORT_GID	Section 29.31.9, on page 2169
VPORT	0x765	QUERY_HCA_VPORT_PKEY	Section 29.31.10, on page 2170
VPORT	0x76F	QUERY_VNIC_ENV	Section 29.34.3, on page 2181
COUNTER	0x770	QUERY_VPORT_COUNTER	Section 29.34.1, on page 2176
COUNTER	0x771	ALLOC_Q_COUNTER	Section 29.35.1, on page 2185
COUNTER	0x772	DEALLOC_Q_COUNTER	Section 29.35.2, on page 2187
COUNTER	0x773	QUERY_Q_COUNTER	Section 29.35.3, on page 2188
COUNTER	0x774	SET_MONITOR_COUNTER	Section 29.35.4, on page 2195
COUNTER	0x775	ARM_MONITOR_COUNTER	Section 29.35.5, on page 2198
QOS	0x780	SET_PP_RATE_LIMIT	Section 29.36.1, on page 2199
QOS	0x781	QUERY_PP_RATE_LIMIT	Section 29.36.2, on page 2201
QOS	0x782	CREATE_SCHEDULING_ELEMENT	Section 29.36.3, on page 2202
QOS	0x783	DESTROY_SCHEDULING_ELEMENT	Section 29.36.4, on page 2207
QOS	0x784	QUERY_SCHEDULING_ELEMENT	Section 29.36.5, on page 2208
QOS	0x785	MODIFY_SCHEDULING_ELEMENT	Section 29.36.6, on page 2210
QOS	0x786	CREATE_QOS_PARA_VPORT	Section 29.36.7, on page 2212
QOS	0x787	DESTROY_QOS_PARA_VPORT	Section 29.36.8, on page 2213
MISC	0x800	ALLOC_PD	Section 29.3.21, on page 1832

Table 1550 - Commands List Sorted by Opcode (Sheet 6 of 11)

Group	Opcode	Command Name	Link to Command Description
MISC	0x801	DEALLOC_PD	Section 29.3.22, on page 1833
MISC	0x802	ALLOC_UAR	Section 29.3.23, on page 1834
MISC	0x803	DEALLOC_UAR	Section 29.3.24, on page 1835
MISC	0x804	CONFIG_INT_MODERATION	Section 29.3.27, on page 1838
MISC	0x805	ACCESS_REGISTER	Section 29.5.1, on page 1857
MISC	0x806	ATTACH_TO_MCG	Section 29.37.1, on page 2214
MISC	0x807	DETACH_FROM_MCG	Section 29.37.2, on page 2216
MISC	0x808	ATTACH_TO_SNIFFER	Section 29.38.1, on page 2217
MISC	0x809	DETACH_FROM_SNIFFER	Section 29.38.2, on page 2219
MISC	0x80A	GET_DROPPED_PACKET_LOG	Section 29.39, on page 2220
MISC	0x50D	MAD_IFC	Section 29.24.3, on page 2101
MISC	0x80B	QUERY_MAD_DEMUX	Section 29.24.1, on page 2096
MISC	0x80C	SET_MAD_DEMUX	Section 29.24.2, on page 2100
MISC	0x80D	NOP	Section 29.43.1, on page 2253
MISC	0x80E	ALLOC_XRCD	Section 29.3.25, on page 1836
MISC	0x80F	DEALLOC_XRCD	Section 29.3.26, on page 1837
MISC	0x810	ACCESS_REG_SPACE - deprecated	Section 29.5.2, on page 1858
MISC	0x811	ACCESS_REG_SPACE_DWORD - deprecated	Section 29.5.3, on page 1859
MISC	0x812	SET_BURST_SIZE	Section 29.3.28, on page 1840
MISC	0x813	QUERY_BURST_SIZE	Section 29.3.29, on page 1841
MISC	0x814	ACTIVATE_TRACER	Section 29.3.30, on page 1842
MISC	0x815	DEACTIVATE_TRACER	Section 29.3.31, on page 1843
MISC	0x816	ALLOC_TRANSPORT_DOMAIN	Section 29.3.32, on page 1844
MISC	0x817	DEALLOC_TRANSPORT_DOMAIN	Section 29.3.33, on page 1845
MISC	0x819	QUERY_DIAGNOSTIC_PARAMS	Section 29.40.2, on page 2224
MISC	0x820	SET_DIAGNOSTIC_PARAMS	Section 29.40.1, on page 2222
MISC	0x821	QUERY_DIAGNOSTIC_CNTRS	Section 29.40.3, on page 2226

Table 1550 - Commands List Sorted by Opcode (Sheet 7 of 11)

Group	Opcode	Command Name	Link to Command Description
MISC	0x822	QUERY_CONG_STATUS	Section 29.41.1, on page 2228
MISC	0x823	MODIFY_CONG_STATUS	Section 29.41.2, on page 2230
MISC	0x824	QUERY_CONG_PARAMS	Section 29.41.3, on page 2232
MISC	0x825	MODIFY_CONG_PARAMS	Section 29.41.4, on page 2243
MISC	0x826	QUERY_CONG_STATISTICS	Section 29.41.5, on page 2247
MISC	0x827	ADD_VXLAN_UDP_DPORT	Section 29.42.1, on page 2251
MISC	0x828	DELETE_VXLAN_UDP_DPORT	Section 29.42.2, on page 2252
MISC	0x829	SET_L2_TABLE_ENTRY	Section 29.19.1, on page 2059
MISC	0x82A	QUERY_L2_TABLE_ENTRY	Section 29.19.2, on page 2060
MISC	0x82B	DELETE_L2_TABLE_ENTRY	Section 29.19.3, on page 2062
MISC	0x830	SET_WOL_ROL	Section 29.3.34, on page 1846
MISC	0x831	QUERY_WOL_ROL	Section 29.3.35, on page 1848
MISC	0x832	ARM_RE-SOURCE_DUMP_EVENT	Section 29.44.7, on page 2265
LAG	0x840	CREATE_LAG	Section 29.4.1, on page 1849
LAG	0x841	MODIFY_LAG	Section 29.4.2, on page 1851
LAG	0x842	QUERY_LAG	Section 29.4.3, on page 1853
LAG	0x843	DESTROY_LAG	Section 29.4.4, on page 1854
LAG	0x844	CREATE_VPORT_LAG	Section 29.4.5, on page 1854
LAG	0x845	DESTROY_VPORT_LAG	Section 29.4.6, on page 1856
TIR	0x900	CREATE_TIR	Section 29.12.1, on page 1949
TIR	0x901	MODIFY_TIR	Section 29.12.2, on page 1950
TIR	0x902	DESTROY_TIR	Section 29.12.3, on page 1952
TIR	0x903	QUERY_TIR	Section 29.12.4, on page 1953
SQ	0x904	CREATE_SQ	Section 29.14.1, on page 1960
SQ	0x905	MODIFY_SQ	Section 29.14.2, on page 1962
SQ	0x906	DESTROY_SQ	Section 29.14.3, on page 1963
SQ	0x907	QUERY_SQ	Section 29.14.4, on page 1964
RQ	0x908	CREATE_RQ	Section 29.15.1, on page 1966

Table 1550 - Commands List Sorted by Opcode (Sheet 8 of 11)

Group	Opcode	Command Name	Link to Command Description
RQ	0x909	MODIFY_RQ	Section 29.15.2, on page 1967
RQ	0x90A	DESTROY_RQ	Section 29.15.3, on page 1969
RQ	0x90B	QUERY_RQ	Section 29.15.3, on page 1969
RQ	0x90C	CREATE_RMP	Section 29.17.1, on page 1980
RQ	0x90D	MODIFY_RMP	Section 29.17.2, on page 1981
RQ	0x90E	DESTROY_RMP	Section 29.17.3, on page 1983
RQ	0x90F	QUERY_RMP	Section 29.17.4, on page 1984
RQ	0x910	SET_DELAY_DROP_PARAMS	Section 29.15.5, on page 1971
RQ	0x911	QUERY_DELAY_- DROP_PARAMS	Section 29.15.6, on page 1972
TIS	0x912	CREATE_TIS	Section 29.13.1, on page 1955
TIS	0x913	MODIFY_TIS	Section 29.13.2, on page 1956
TIS	0x914	DESTROY_TIS	Section 29.13.3, on page 1958
TIS	0x915	QUERY_TIS	Section 29.13.4, on page 1959
RQT	0x916	CREATE_RQT	Section 29.16.1, on page 1974
RQT	0x917	MODIFY_RQT	Section 29.16.2, on page 1975
RQT	0x918	DESTROY_RQT	Section 29.16.3, on page 1977
RQT	0x919	QUERY_RQT	Section 29.16.4, on page 1978
FLOW TABLE	0x92f	SET_FLOW_TABLE_ROOT	Section 29.18.4, on page 1995
FLOW TABLE	0x930	CREATE_FLOW_TABLE	Section 29.18.1, on page 1986
FLOW TABLE	0x931	DESTROY_FLOW_TABLE	Section 29.18.3, on page 1993
FLOW TABLE	0x932	QUERY_FLOW_TABLE	Section 29.18.5, on page 1998
FLOW TABLE	0x933	CREATE_FLOW_GROUP	Section 29.18.6, on page 2000
FLOW TABLE	0x934	DESTROY_FLOW_GROUP	Section 29.18.7, on page 2018
FLOW TABLE	0x935	QUERY_FLOW_GROUP	Section 29.18.8, on page 2020

Table 1550 - Commands List Sorted by Opcode (Sheet 9 of 11)

Group	Opcode	Command Name	Link to Command Description
FLOW TABLE	0x936	SET_FLOW_TABLE_ENTRY	Section 29.18.9, on page 2023
FLOW TABLE	0x937	QUERY_FLOW_TABLE_ENTRY	Section 29.18.10, on page 2034
FLOW TABLE	0x938	DELETE_FLOW_TABLE_ENTRY	Section 29.18.11, on page 2036
FLOW TABLE	0x939	ALLOC_FLOW_COUNTER	Section 29.18.12, on page 2038
FLOW TABLE	0x93a	DEALLOC_FLOW_COUNTER	Section 29.18.13, on page 2040
FLOW TABLE	0x93b	QUERY_FLOW_COUNTER	Section 29.18.14, on page 2041
FLOW TABLE	0x93c	MODIFY_FLOW_TABLE	Section 29.18.2, on page 1991
FLOW TABLE	0x93d	ALLOC_PACKET_REFORMAT_CONTEXT	Section 29.18.15, on page 2043
FLOW TABLE	0x93e	DEALLOC_PACKET_REFORMAT_CONTEXT	Section 29.18.16, on page 2046
FLOW TABLE	0x93f	QUERY_PACKET_REFORMAT_CONTEXT	Section 29.18.17, on page 2047
FLOW TABLE	0x940	ALLOC_MODIFY_HEADER_CONTEXT	Section 29.18.18, on page 2048
FLOW TABLE	0x941	DEALLOC_MODIFY_HEADER_CONTEXT	Section 29.18.19, on page 2054
FLOW TABLE	0x942	QUERY_MODIFY_HEADER_CONTEXT	Section 29.18.20, on page 2055
FLOW TABLE	0x943	MUST_NOT_USE	
FPGA	0x960	FPGA_CREATE_QP	Section 29.20.1, on page 2063
FPGA	0x961	FPGA_MODIFY_QP	Section 29.20.2, on page 2065
FPGA	0x962	FPGA_QUERY_QP	Section 29.20.3, on page 2066
FPGA	0x963	FPGA_DESTROY_QP	Section 29.20.4, on page 2067
FPGA	0x964	FPGA_QUERY_QP_COUNTERS	Section 29.20.5, on page 2068
AFU	0x970	CREATE_AFU	Section 29.21.1, on page 2070

Table 1550 - Commands List Sorted by Opcode (Sheet 10 of 11)

Group	Opcode	Command Name	Link to Command Description
AFU	0x971	QUERY_AFU	Section 29.21.2, on page 2073
AFU	0x972	DESTROY_AFU	Section 29.21.3, on page 2074
CAPI	0x980	CREATE_CAPI_PEC	Section 29.22.1, on page 2075
CAPI	0x981	QUERY_CAPI_PEC	Section 29.22.2, on page 2077
CAPI	0x982	DESTROY_CAPI_PEC	Section 29.22.3, on page 2078
OBJECT	0xA00	CREATE_GENERAL_OBJECT	Section 29.23.1, on page 2081
OBJECT	0xA01	MODIFY_GENERAL_OBJECT	Section 29.23.2, on page 2084
OBJECT	0xA02	QUERY_GENERAL_OBJECT	Section 29.23.3, on page 2085
OBJECT	0xA03	DESTROY_GENERAL_OBJECT	Section 29.23.4, on page 2086
UCTX	0xA04	CREATE_UCTX	Section 29.23.5, on page 2087
UCTX	0xA05	QUERY_UCTX	Section 29.23.6, on page 2088
UCTX	0xA06	DESTROY_UCTX	Section 29.23.7, on page 2090
UMEM	0xA08	CREATE_UMEM	Section 29.23.8, on page 2090
UMEM	0xA09	QUERY_UMEM	Section 29.23.9, on page 2093
UMEM	0xA0A	DESTROY_UMEM	Section 29.23.10, on page 2094
USER_- GEN- ERAL	0xB00- 0xDFF	SAVED_FOR_USER_GENERAL_COMMANDS	
USER_- GEN- ERAL	0xB00	SYNC_STEERING	Section 29.18.21, on page 2057
USER_- GEN- ERAL	0xB01	SEND_NVMF_CC	Section 29.10.17, on page 1936
USER_- GEN- ERAL	0xB02	QUERY_NVMF_CC_RESPONSE	Section 29.10.18, on page 1938
USER_- GEN- ERAL	0xB03	QUERY_EMULATED_FUNCTIONS_INFO	Section 29.10.19, on page 1940
REGEXP	0xB04	SET_REGEXP_PARAMS	Section 29.11.1, on page 1942
REGEXP	0xB05	QUERY_REGEXP_PARAMS	Section 29.11.2, on page 1944
REGEXP	0xB06	SET_REGEXP_REGISTER	Section 29.11.3, on page 1946

Table 1550 - Commands List Sorted by Opcode (Sheet 11 of 11)

Group	Opcode	Command Name	Link to Command Description
REGEXP	0xB07	QUERY_REGEX_REGISTER	Section 29.11.4, on page 1947
XRQ	0xB08	USER_QUERY_XRQ_D-C_PARAMS_ENTRY	Section 29.27.5, on page 2122
XRQ	0xB09	USER_SET_XRQ_D-C_PARAMS_ENTRY	Section 29.27.6, on page 2124
XRQ	0xB0A	USER_QUERY_XRQ_ERROR_PARAMS	Section 29.27.7, on page 2125
XRQ	0xB0B	USER_RELEASE_XRQ_ERROR	Section 29.27.8, on page 2128
EMC	0x8000	QUERY_RSV_RESOURCES	Section 29.44.1, on page 2254
EMC	0x8001	QUERY_MTT	Section 29.44.4, on page 2259
EMC	0x8002	CREATE_MTT	Section 29.44.2, on page 2256
EMC	0x8003	DESTROY_MTT	Section 29.44.3, on page 2258
EMC	0x8004	ACCESS_CR	Section 29.44.5, on page 2260
EMC	0x8005	MODIFY_SCHED_QUEUE	Section 29.44.6, on page 2261
EMC	0x8006	QUERY_SCHED_QUEUE	Section 29.44.6, on page 2261

29.2 Return Status Summary

The following table summarizes the return status enumerators for all the commands.

Table 1551 - Return Status Summary

Category	Value	Name	Description
General	0x00	OK	Command execution succeeded
	0x01	INTERNAL_ERR	Internal error (for example bus error) occurred while processing command
	0x02	BAD_OP	Operation/command not supported or opcode modifier not supported
	0x03	BAD_PARAM	Parameter not supported, parameter out of range, reserved not equal 0
	0x04	BAD_SYS_STATE	System was not enabled or bad system state
	0x05	BAD_RESOURCE	Attempt to access reserved or unallocated resource, or resource in inappropriate status. for example, not existing CQ when creating QP/SQ/RQ
	0x06	RESOURCE_BUSY	Requested resource is currently executing a command. No change in any resource status or state. i.e. command just not executed.
	0x08	EXCEED_LIM	Required capability exceeds device limits
	0x09	BAD_RES_STATE	Resource is not in the appropriate state or ownership
	0x0A	BAD_INDEX	Index out of range (might be beyond table size or attempt to access a reserved resource)
	0x0F	NO_RESOURCES	Command was not executed because of lack of resources (for example ICM pages). This is unrecoverable situation from driver point of view
	0x50	BAD_INPUT_LEN	Bad command input len
	0x51	BAD_OUTPUT_LEN	Bad command output len
Applies for resources which state can be changed via command i/f: QP/RQ/SQ/TIR/TIS	0x10	BAD_RESOURCE_STATE	Attempt to modify a Resource (RQ/SQ/TIR/TIS/QPs) which is not in the presumed state

Table 1551 - Return Status Summary (Continued)

Category	Value	Name	Description
MAD	0x30	BAD_PKT	Bad management packet (silently discarded)
CQ	0x40	BAD_SIZE	More outstanding CQEs in CQ than new CQ size
EMC	0x60	PENDING - COMPLETION	This QP is currently pending device completion

Note: Hardware is not required to check all parameters of the command interface. To speed up command execution in hardware, some of the checks may not be implemented by the adapter device, and thus not all return status values are applicable. The basic assumption is that software performs all appropriate checks prior to executing the command. Errors that may not be returned include: BAD_SYS_STATE, BAD_RES_STATE, BAD_INDEX, INTERNAL_ERR, etc.

Note: Not all return statuses apply to each command.

29.3 Initialization and General Commands

Once HCA has booted and undergone the PCI enumeration, the driver should perform the steps defined in the initialization chapter (see “HCA Driver Start-up” on page 244).

To take down the HCA, the driver should follow the steps described in Section 7.2, “HCA Driver Start-up,” on page 244.

The following are initialization and general commands that can be executed from the command interface.

Table 1552 - Initialization and General Commands

Mnemonic	Command	Description	Reference
QUERY_PAGES	Query device free memory pool status	Returns the status of the free memory page pool	Section 29.3.1, on page 1660
MANAGE_PAGES	Deliver / return memory pages to the device	Passes additional memory pages for the device usage	Section 29.3.2, on page 1662
QUERY_HCA_CAP	Query device capabilities	Returns the device limits and the capabilities supported	Section 29.3.4, on page 1667
SET_HCA_CAP	Set device capabilities	Sets the device limits and the capabilities supported	Section 29.3.5, on page 1803
QUERY_ADAPTER	Query adapter	Queries card/board parameters and properties	Section 29.3.12, on page 1816
INIT_HCA	INIT HCA	Initiates and opens the HCA	Section 29.3.13, on page 1820

Table 1552 - Initialization and General Commands (Continued)

Mnemonic	Command	Description	Reference
TEAR-DOWN_HCA	Tear-down HCA	Releases all HCA allocated resources	Section 29.3.14, on page 1821
ALLOC_PD	Allocate protection domain	Allocates PD	Section 29.3.21, on page 1832
DEALLOC_PD	De-allocate protection domain	De-allocates PD	Section 29.3.22, on page 1833
ALLOC_UAR	Allocate UAR	Allocates UAR	Section 29.3.23, on page 1834
DEALLOC_UAR	De-allocate UAR	De-allocates UAR	Section 29.3.24, on page 1835
ALLOC_XRCD	Allocate XRC domain	Allocates XRCD	Section 29.3.25, on page 1836
DEALLOC_XRCD	De-allocate XRC domain	De-allocates XRCD	Section 29.3.26, on page 1837
CONFIG_INT_MODERATION	Configure interrupt moderation	Sets a maximal interrupt frequency	Section 29.3.27, on page 1838
HW_HEALTH_CHECK	Hardware health checking	Checks internal hardware entities for health	
QUERY_FUNC	Query Function	Query PCI function attributes	

29.3.1 QUERY_PAGES - Query Device Free Memory Pool Status

The QUERY_PAGES command returns the status of the free memory page pool. A positive number indicates that the function has spare pages that can be reclaimed by the host. A given number indicates that the function needs more pages, which should be provided by software using the MANAGE_PAGES command. The value is the number of spare / needed 4KB pages.

Table 1553 - QUERY_PAGES Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				

Table 1553 - QUERY_PAGES Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
embedded_cpu_function																function_id																08h
																																0Ch

Table 1554 - QUERY_PAGES Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Opcode modifier 0x1: boot_pages 0x2: init_pages 0x3: regular_pages	
08h	31	embed-ded_cpu_function	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on external Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU INIT_SEGMENT.embedded_cpu==1	
	15:0	function_id	function_id of the device to enable. 0: when accessing my function or when ECPF access PF function. others: when vhca_group_manager (HCA - CAP.vhca_group_manager==1) accesses another function in its group ~~or on behalf of it(embed_on_behalf==1).~~	

Table 1555 - QUERY_PAGES Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 1555 - QUERY_PAGES Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
embedded cpu function																function_id																08h
		num_pages																														0Ch

Table 1556 - QUERY_PAGES Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		
08h	31	embed- ded_cpu_func- tion	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on external Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU INIT_SEGMENT.embedded_cpu==1	
	15:0	function_id	function_id of the device to enable. 0: when accessing my function or when ECPF access PF function. others: when vhca_group_manager (HCA - CAP.vhca_group_manager ==1) accesses another function in its group ~~or on behalf of it (cmd_on_behalf=1).~~	
0Ch	31:0	num_pages	Number of pages the device requires from the driver (positive numbers means driver should provide pages. Negative num- ber means the driver should reclaim pages).	

29.3.2 MANAGE_PAGES - Driver Delivers Memory Pages for the Device Usage or Returns Pages

The MANAGE_PAGES command passes 4KB physical pages to the HCA or returns them to the host. This memory is used by the HCA to store its various contexts. MANAGE_PAGES must be used prior to INIT_HCA to provide the device with the initial memory it requires for boot. MANAGE_PAGES should be executed once, per event, (per function_id) in response to Page Request Event. Physical memory passed in this command must be pinned. This command also

arms the Page Request Event. This implies that once Page Request Event is generated for a function_id, it will not be generated again before the driver executes this command.

Driver will execute `MANAGE_PAGES` with opcode modifier = 0 if it was required to provide pages but is unable to (for example, failure to allocate memory).

Table 1557 - `MANAGE_PAGES` Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
embedded cpu function																function_id												08h				
	input_num_entries																0Ch															
	pas[0]																10h-14h															
	pas[1]																18-1Ch															
...																...																

Table 1558 - `MANAGE_PAGES` Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod	Opcode modifier 0x0: ALLOCATION_FAIL - SW cannot give pages. No mailbox is valid. 0x1: ALLOCATION_SUCCESS - SW gives pages to HCA. Input parameter and input mailbox are valid. 0x2: HCA_RETURN_PAGES - SW requests to return pages from HCA back to the host. Input parameter, output parameter and output mailbox are valid.	

Table 1558 - MANAGE_PAGES Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	embed- ded_cpu_func- tion	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on external Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU INIT_SEGMENT.embedded_cpu==1	
	15:0	function_id	function_id of the device to enable. 0: when accessing my function or when ECPF access PF function. others: when vhca_group_manager (HCA_- CAP.vhca_group_manager ==1) accesses another function in its group ~~or on behalf of it(embed_on_behalf==1).~~	
0Ch	31:0	input_num_en- tries	Number of valid PAS entries	
10h- ...	64	pas[...] (See Table 1561, “Physi- cal_Address_Struc- ture (PAS) Layout,” on page 1665)		

Table 1559 - MANAGE_PAGES Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
output_num_entries																																08h
																																0Ch
pas[0]																																10h-14h
pas[1]																																18h-1Ch
...																																...

Table 1560 - MANAGE_PAGES Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		
08h	31:0	output_num_entries	Number of returned PAS entries	
10h- ...	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)		

The MANAGE_PAGES command uses the following Physical Address Structure (PAS) layout data structure.

Table 1561 - Physical Address Structure (PAS) Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pa_h																															00h	
pa_l																															04h	

Table 1562 - Physical Address Structure (PAS) Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	pa_h	Physical Address [63:32]	
04h	31:1 2	pa_l	Physical Address [31:12]	

29.3.3 SET_TUNNELED_OPERATION_PARAMS - Set Tunneled Operation Parameters

This command is used to configure the required parameters needed to support and handle Tunneled operations (See TBD). This command should be performed in the initialization flow as mentioned in Section 7.2, “HCA Driver Start-up,” on page 244

Table 1563 - SET_TUNNELED_OPERATION_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h

Table 1563 - SET_TUNNELED_OPERATION_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																op_mod												04h				
asn_match_mask																asn_match_value												08h				
																														0C-3Ch		

Table 1564 - SET_TUNNELED_OPERATION_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod		
08h	31:16	asn_match_mask	Mask of the MSB address, to determine which bits of asn_match_value are valid (determine the asn_match size).	
	15:0	asn_match_value	MSB of the address, used to identify tunneled operation	

Table 1565 - SET_TUNNELED_OPERATION_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																												04h				
																				log_response_bar_size								08h				
																												0Ch				
response_bar_address																				10h-14h								18h-1Ch				

Table 1566 - SET_TUNNELED_OPERATION_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		
0Ch	31:0	log_response_bar_size	log (base 2) of the size of the HCA response bar.	
10h- 14h	64	response_bar_address	Address of response BAR on the HCA. The host sends responses to tunneled operations toward this BAR	

29.3.4 QUERY_HCA_CAP – Query Device Capabilities

The QUERY_HCA_CAP command returns the device limits and the capabilities supported. Some of the device capabilities can be configured through the NVMEM attached to the adapter device and some reflect actual HW/FW capabilities.

Table 1567 - QUERY_HCA_CAP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
other function																function_id												08h				
																																0Ch

Table 1568 - QUERY_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:1 6	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 1568 - QUERY_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod	Bit[0] indicates Maximum or Current capabilities 0x0: Maximum 0x1: Current Bits[15:1] indicates Capability Type 0x0: General_Device_Capabilities 0x1: Ethernet_Offload_Capabilities 0x2: ODP_Capabilities 0x3: Atomic_Capabilities 0x4: RoCE_Capabilities 0x5: IPoIB_Basic_Offload_Capabilities 0x6: IPoIB_Enhanced_Offload_Capabilities 0x7: NIC_Flow_Table_Capabilities 0x8: Eswitch_Flow_Table_Capabilities 0x9: Eswitch_Capabilities 0xB: Vector_Calc_Capabilities 0xC: Quality_of_Service_Capabilities 0xD: Debug_Capabilities 0xE: NVMe-oF_Capabilities 0xF: Device_Memory_Capabilities 0x10: NVME_Device_Emulation_Capabilities 0x11: TLS_Capabilities 0x12: DPP_Capabilities 0x13: VDPA_EMULATION_Capabilities 0x14: EVENT_Capabilities 0x15: IPSEC_Capabilities 0x16: VIRTIO_NET_EMULATION_Capabilities 0x17: VIRTIO_BLK_EMULATION_Capabilities 0x18: HOTPLUG_Capabilities 0x19: NVMEOTCP_Capabilities 0x1A: CRYPTO_Capabilities 0x1B: WQE_BASED_FLOW_TABLE_Capabilities	
08h	31	other_function	If set, indicates accessing capabilities of other function identified by function_id. Valid only for vhca resource manager (HCA_CAP.vhca_resource_manager==1)	
	15:0	function_id	function_id of vHCA: 0: When accessing my vHCA. 0xffff: Access all function_ids. others: Function_id of specific vHCA used by vhca_resource_manager (HCA_CAP.vhca_resource_manager==1).	

Table 1569 - QUERY_HCA_CAP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
capability																																10h-100Ch

Table 1570 - QUERY_HCA_CAP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 1570 - QUERY_HCA_CAP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-100Ch	32768	capability	Capability structure according to the op_mod. op_mod[0:0] indicates Maximum or Current capabilities 0x0: Maximum 0x1: Current op_mod[15:1] indicates Capability Type 0x0: General Device Capabilities. See Table 1571, “HCA Capabilities Layout,” on page 1671. 0x1: Ethernet Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734 0x2: ODP Capabilities. See Table 1573, “ODP Capabilities Layout,” on page 1726. 0x3: Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728. 0x4: RoCE Capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732. 0x5: IPoIB Basic Offload Capabilities. 0x6: IPoIB Enhanced Offload Capabilities. 0x7: NIC Flow Table Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742 0x8: E-switch Flow Table Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748 0x9: E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768 0xB: Vector Calc Capabilities. See Table 1595, “Vector Calc Capabilities Layout,” on page 1772 0xC: Quality of Service Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773 0xD: Debug Capabilities. See Table 1599, “Debug Capabilities Layout,” on page 1777 0xE: NVMe-oF Capabilities. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780 0xF: Device_Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785 0x10: NVME Device Emulation Capabilities. See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787 0x11: TLS Capabilities. See Table 1609, “TLS Capabilities Layout,” on page 1789 0x12: DPP Capabilities. See Table 1613, “DPP Capabilities Layout,” on page 1792 0x13: VDPA_EMULATION Capabilities. 0x14: EVENT_Capabilities. See See Table 1617, “Event Capabilities Layout,” on page 1795 0x15: IPSEC Capabilities. See Table 1611, “IPSEC Capabilities Layout,” on page 1790 0x16: VIRTIO_NET_EMULATION_Capabilities. 0x17: VIRTIO_BLK_EMULATION_Capabilities. See Table 1615, “VIRTIO_Emulation Capabilities Layout,” on page 1793 0x18: HOTPLUG_Capabilities. See Table 1619, “Hotplug Capabilities Layout,” on page 1798 0x19: NVMEOTCP_Capabilities. See Table 1623, “NVMEoTCP Capabilities Layout,” on page 1801 0x1A: CRYPTo_Capabilities	

Table 1571 - HCA Capabilities Layout

[illegible]

Table 1571 - HCA Capabilities Layout

Offset	1Ch	20h	24h
0	log_max_eq	log_max_klm_list_size	log_max_ra_res_dc
1			
2			
3	fast teardown		
4	resources on nvme emulation manager		
5	dpp	null mkey	
6	dump fill mkey	unr extended translation offset	
7	repeated mkey		
8	terminate scatter list mkey		
9	nvme device emulation manager		
10	hotplug manager		
11	m_vhea mk		
12	m_pci port		
13	as notify	hard coded 1	
14	tunneled atomic	force teardown	
15			
16	log_max_mkey	log_max_mrw_sz	log_max_ra_req_dc
17			
18			
19			
20			
21			
22	relaxed ordering read		resources on virtio blk emulation manager
23	relaxed ordering write	fixed buffer size	resources on virtio net emulation manager
24			
25			
26			
27			
28			
29	log_max_eq_sz	max_indirection	
30			
31			

Table 1571 - HCA Capabilities Layout

Offset	28h	2Ch	30h	34h	
0	log_max_ra_res_qp	gid_table_size	pkey_table_size	num_ports	
1					
2					
3					
4					
5					
6	log_max_ra_req_qp				
7					
8					
9					
10					
11					
12					
13					
14					
15					
16	log_max_ra_req_qp	qcam_reg qcwcam_reg sbcam_reg vnic_env_int_rq_oob gen3_cc_negotiation rts2rts_qp_dscp rts2rts_qp_counters_set_id rnr_nak_q_counters rts2rts_qp_rmp pkey_table_size_ro gid_table_size_ro	max_qp_cnt	local_ca_ack_delay pcam_reg mcam_reg device_memory eswitch_manager nic_flow_table ets	
17					
18					
19					
20					
21					
22	roce_acl				
23	sig_crc32c				
24	sig_crc64_xp10				
25	release_all_pages				
26					
27	cc_roce_een_rp_burst_decouple				
28	cc_roce_een_rp_dynamic_g				
29	cc_roce_een_rp_dynamic_ai				
30	cc_roce_een_rp_dynamic_rtt				
31	cc_roce_een_rp_classify_mode				

Table 1571 - HCA Capabilities Layout

Offset	38h	3Ch	40h	44h	48h
0	wol_p	cqe_version	extended_retry_count	rc	log_pg_sz
1	wol_u		atomic	uc	
2	wol_m		roce	ud	
3	wol_b		eth_net_offloads	xrc	
4	wol_a	wqe_based_flow_table_update	qos	dc_v2	log_max_dc_c_nak_qps
5	wol_g	stat_rate_200	dct	initiator_src_dct	
6	wol_s	flow_group_type_hash_split	rf	rts2rts_primary_sl	
7	ib_port_sniffer	init2_lag_tx_port_affinity	tph	set_deth_sqpn	
8	rol_g	log_max_extended_rnr_retry	sho	pkv	log_pg_sz
9	rol_s		bsf_in_create_mkey	qkv	
10	fpga		scr_data_cqe	initiator_src_dct_in_cqe	
11	afu_owner		wq_signature	ipoib_enhanced_pkey_change	
12	multi_prio_sq	init2tr_drain_sigerr	e_psv	ib_cyclic_striding_wq	uar_sz
13	general_notification_event		sigerr_cqe	qp_packet_based	
14	debx		cmdif_checksum	imaicl	
15	temp_warn_event		pci_sync_for_fw_update_event	umr_ptr_rlkey	
16	max_tc	sig_block_4048	drain_sigerr	vector_calc	uar_sz
17			dc_cnak_trace	apm	
18			dc_connect_qp	atm	
19			dc_req_scr_data_cqe	cd	
20	traffic_fast_control	stat_rate_support	umr_fence	cq_period_start_from_cqe	uar_sz
21	multi_path_rc_rdma		scqe_break_moderation		
22	multi_path_dc_rdma		exponential_backoff		
23	multi_path_xrc_rdma		block_lb_mc		
24	log_max_msg		umr_modify_entity_size_disabled	pg	
25			repeated_block_disabled	cq_eq_remap	
26		ib_link_list_striding_wq			
27		ipoib_basic_offloads	cq_invalidate		
28	log_max_msg	ipoib_enhanced_offloads	cq_period_mode_modify	uar_4k	
29		Internal_keep_it_reserved	cq_moderation		
30		eth_striding_wq	cq_resize		
31		reserved	compact_address_vector		cq_oi

Table 1571 - HCA Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
log_bf_reg_size																						lag_dct		lag_tx_port affinity		lag native fdb selection		must_be_0		lag_master		num_lag_ports				04Ch
num_of_diagnostic_counters																								max_wqe_sz_sq										50h		
																								max_wqe_sz_rq										54h		
max_flow_counter_31_16																								max_wqe_sz_sq_dc										58h		
							max_qp_mcg																			5Ch										
mlnx_tag_ethertype																flow_counter_bulk_alloc						log_max_mcg						60h								
			log_max_transport_domain								log_max_pd										log_max_xrxd				64h											
nic_receive_steering_discard		receive_discard_vport_down		transmit_discard_vport_down		eq overrun count		invalid command count		quota exceeded count		must_not_use		log_max_flow_counter_bulk							max_flow_counter_15_0										68h					
modify_tis		flow_counters_dump		log_max_rq								log_max_sq								log_max_tir								log_max_tis				6Ch				
basic_cyclic_rev_wqe		mem_rq_rmp		log_max_rmp								log_max_rqt								log_max_rqt_size								log_max_tis_per_sq				70h				

Table 1571 - HCA Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
ext_stride_num_range			log_max_stride_sz_rq								log_min_stride_sz_rq							log_max_stride_sz_sq							log_min_stride_sz_sq				74h						
hairpin			log_max_hairpin_queues					hairpin_ib_raw			hairpin_eth2ipoib			hairpin_ipoib2eth			log_max_hairpin_wq_data_sz							log_max_hairpin_num_packets							log_max_wq_sz				78h
nic_vport_change_event		disable_local_lb_uc		disable_local_lb_uc		log_min_hairpin_wq_data_sz		system_image_guid_modifiable					log_max_vlan_list							log_max_current_mc_list							log_max_current_uc_list				7Ch				
general_obj_types																														80h-84h					
			steering_format_version					create_qp_start_hint																							88h				
must_be_0		ats		log_max_uctx					aes_xts		crypto		ipsec_offload			log_max_umem				max_num_eqs															8Ch
		tls_tx		tls_rx		log_max_l2_table													log_uar_page_sz															90h	
c																														94h					
device_frequency_mhz																														98h					
device_frequency_khz																														9Ch					

Table 1571 - HCA Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
																															A0h																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
			capi	nvmmf target offload																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
num_of_uars_per_page																																A4h																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
flex_parser_protocols																																A8h																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
max_geneve_tlv_options												max_- geneve_tlv_op- tion_data_len								log_max_guar- anteed_connek- tions								log_max_dct_- connections				ACh																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
log_max_atomic_size_qp												log_max_d- ci_stream_chan- nels								log_max_d- ci_er- rored_streams				log_max_atomic_size_dc				B0h																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
max_multi_user_group_size																			log_max_dek																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								

Table 1571 - HCA Capabilities Layout

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
D0h	max_num_sf_partitions								log_min_sf_size																log_max_sf				sf_set_partition sf																			
D4h	uctx_permission																																															
D8h	flex_parser_id_outer_first_mpls_over_udp_label								flex_parser_id_outer_first_mpls_over_gre								flex_parser_id_icmpv6_dw0								flex_parser_id_icmpv6_dw1								flex_parser_id_icmp_dw1								flex_parser_id_geneve_tlv_option_0							
DCh	sf_base_id																max_num_match_definer																															
E0h	num_total_dynamic_vf_msix																																															
E4h	dynamic_msix_table_size																																															
E8h	max_dynamic_vf_msix_table_size																min_dynamic_vf_msix_table_size																															
ECh																																																
F0h-F4h	vhca_tunnel_commands																																															
F8h-FCh	match_definer_format_supported																																															

Note: Empty entry in “Access of current capabilities” indicates that field is Read Only. Access Entry with RW indicates that field is Read Write.

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
00h	31	access_other_hca_roce	If set, vhca group manager is capable to control “roce” field in HCA_CAP for another vHCA under its responsibility. See Section 29.3.6, “QUERY_OTHER_HCA_CAP - Query Other HCA Capabilities,” on page 1807.		No [FW]
	0	vhca_resource_manager	vHCA resource manager. If set, vhca resource manager is capable to control capabilities of other vHCAs under its responsibility.		No [FW]
04h	15:0	vhca_id	Virtual HCA identifier.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
10h	31:24	log_max_srq_sz	Log (base 2) of the maximum number of WQEs allowed on the SRQ/XRC_SRQ.		Yes [TRUST]
	23:16	log_max_qp_sz	Log (base 2) of the maximum number of WQEs allowed on the RQ or the SQ of a QP.		Yes [TRUST]
	15	event_cap	If set, Event Capabilities structure is supported. See Table 1617, "Event Capabilities Layout," on page 1795		No [FW]
	9	nvmeotcp	If set, NVMeoTCP offload is supported. For detailed capabilities see Table 1617, "Event Capabilities Layout," on page 1795		Yes [TRUST]
	8	pcie_hanged	If set, pci_hanged report is supported. See Section 27.7, "PCIe Hanged Event," on page 1061		No [FW]
	7	prio_tag_required	If set, the system requires to always have prio tag on the network. SW should set the required flow table rules to ensure that. Enabled by NV_SW_OFFLOAD_CONFIG.prio_tag_required_en. <i>[Internal] This field is added as a solution for verizon where Pop vlan not supported in Tx and push not supported in Rx. Reported only to eswitch manager that should set the required flow table rules to ensure VFs traffic with prio tag.</i>		No [FW]
	6	wqe_index_ignore_cap	When set, wqe_index in the WQE Control element is ignored by the HCA. Notes: - This field is NOT ignored in BlueFlame. - It is recommended to set the right wqe_index in case wq_signature is enabled.		No [FW]
	5	dynamic_qp_allocation	If set, indicates that the device supports dynamic allocation of resources for QPs. <i>[Internal] (Supported starting from ConnectX-5).</i>		No [FW]
	4:0	log_max_qp	Log (base 2) of the maximum number of QPs supported	RW	Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
14h	31	regex	If set, REGEX is supported. SW can use MMO WQE of type REGEXP.		Yes
	30	regex_params	If set, QUERY_REGEX_PARAMS and SET_REGEX_PARAMS are supported.		Yes
	29	regex_alloc_on-behalf_umem	If set, ECPF should allocate regex umem used by external host and only when external host has capability of regex_params. <i>[Internal] WA for BF2 that will be fixed in BF3. i.e. this cap should be reported as 1 for ECPF in BF2 and 0 in BF3.</i>		No [FW]
	<u>28</u>	<u>ece</u>	<u>If set, ECE Enhanced Connection Establishment (ECE) in QP commands is supported.</u>		<u>No [FW]</u>
	27:24	regex-p_num_of_engines	Number of REGEXP engines supported by the device. Valid only when regex==1.		Yes
	20:16	log_max_srq	Log (base 2) of the maximum number of SRQs/XRC_S-RQs supported		Yes [QUOTA]
	12:8	regex-p_log_crspace_size	Log(base 2) of REGEXP CRSPACE size that can be accessed by QUERY_REGEX_REGISTER and SET_REGEX_REGISTER commands. 0 means REGEXP REGISTERS can not be accessed by the commands. Valid only when regex==1.		Yes
	7	selective_repeat	If set, sr_enable in QP and DCT is supported		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
18h	31:24	max_sgl_for_optimized_performance	Maximum scatter gather list for optimized performance. 0 means invalid (no recommended value)		No [FW]
	23:16	log_max_cq_sz	Log (base 2) of the maximum CQEs allowed in a CQ		No [FW]
	15	relaxed_ordering_write_umr	If set, relaxed_ordering_write can be modified by UMR. See Table 362, "MKey Context Format," on page 629.		Yes [Trust]
	14	relaxed_ordering_read_umr	If set, relaxed_ordering_read can be modified by UMR. See Table 362, "MKey Context Format," on page 629.		Yes [Trust]
	6	virtio_net_device_emulation_manager	If set, this vHCA is allowed to manage VIRTIO NET device emulation. Detailed capabilities reported in VIRTIO Device emulation Capabilities (See Table 1615, "VIRTIO_Emulation Capabilities Layout," on page 1793)		No [FW]
	5	virtio_blk_device_emulation_manager	If set, this vHCA is allowed to manage VIRTIO BLK device emulation. detailed capabilities reported in VIRTIO Device emulation Capabilities (See Table 1615, "VIRTIO_Emulation Capabilities Layout," on page 1793)		No [FW]
	4:0	log_max_cq	Log (base 2) of the maximum number of CQs supported		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
1Ch	31:24	log_max_eq_sz	Log (base 2) of the maximum EQEs allowed in an EQ		No [FW]
	23	relaxed_ordering_write	If set, relaxed_ordering_write can be set in Mkey Context when creating Mkey. See Table 362, "MKey Context Format," on page 629.		Yes [Trust]
	22	relaxed_ordering_read	If set, relaxed_ordering_read can be set in Mkey Context when creating Mkey. See Table 362, "MKey Context Format," on page 629.		Yes [Trust]
	21:16	log_max_mkey	Log (base 2) of the maximum number of data MKey entries (the number of Regions/Windows)		Yes [QUOTA]
	15	tunneled_atomic	If set, tunneled_atomic in Mkey Context is supported. See Table 363, "MKey Context Fields," on page 631		No [FW]
	14	as_notify	If set, AS_NOTIFY is supported. See TBD		No [FW]
	13	m_pci_port	If set, Multi PCI Port is supported. See Section 14.9, "Multi PCI Port," on page 857		Yes [Trust]
	12	m_vhca_mk	If set, Multi vHCA Mkey is supported. See Section 14.8, "Multi vHCA Mkey," on page 856		Yes [Trust]
	11	hotplug_manager	If set, this vHCA is allowed to manager (add/remove) hotplug devices. Detailed capabilities reported in NVME Device emulation Capabilities (See Table 1619, "Hotplug Capabilities Layout," on page 1798)		No [FW]
	10	nvme_device_emulation_manager	If set, this vHCA is allowed to manage NVME device emulation. detailed capabilities reported in NVME Device emulation Capabilities (See Table 1607, "NVME Device Emulation Capabilities Layout," on page 1787)		No [FW]
	9	terminate_scatter_list_mkey	If set, <i>terminate_scatter_list_mkey</i> is reported in QUERY_SPECIAL_CONTEXT command. See Section 29.6.5, "QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers," on page 1871.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
1Ch	8	repeated_mkey	If set, repeated_mkey is reported in QUERY_SPECIAL_CONTEXT command. See Section 29.6.5, “QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers,” on page 1871.		No [FW]
	7	dump_fill_mkey	If set, dump_fill_mkey is reported in QUERY_SPECIAL_CONTEXT command. See Section 29.6.5, “QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers,” on page 1871.		No [FW]
	6	dpp	If set, Direct Packet Placement is supported. To get detailed DPP capabilities, software should query DPP cap See Section 29.3.4.16, “DPP Capabilities,” on page 1791.		No [FW]
	5	resources_on_nvme_emulation_manager	If set, allocating resources on NVMe emulation manager (instead of emulated device by a tunnel) is supported. This is done by setting NVME_DEVICE_MULATION_OBJECT.resources_on_emulation_manager=1.		No [FW]
	4	fast_teardown	If set, fast teardown is supported. See Section 7.3.1, “Fast Teardown,” on page 245		No [FW]
	3:0	log_max_eq	Log (base 2) of the maximum number of EQs. Note: SW should use this capability field only in case HCA_CAP.num_of_eqs==0. Otherwise, num_of_eqs should be used.		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
20h	31:24	max_indirection	Maximum level of Mkey indirection supported		Yes [QUOTA]
	23	fixed_buffer_size	If set, Mkey access_mode of type fixed buffer size is supported. See Table 363, “MKey Context Fields,” on page 631		TBD - check min value
	22:16	log_max_mrw_sz	Log (base 2) of the maximum size of a Memory Region/Window		No [Unlimited]
	15	force_tearardown	If set, force tearardown of HCA is supported using TEARDOWN_HCA command. See Section 29.3.14, “TEARDOWN_HCA – Tear-down HCA,” on page 1821.		No [FW]
	14	hard_coded_1	Replaced by fast_tearardown due to FW bug. FW must report 1 in this internal field		No [FW]
	13:8	log_max-bsf_list_size	Log (base 2) of the maximum bsf entries list (in MKey)		Yes [QUOTA]
	7	umr_extended_translation_offset	If set, UMR Control segment translation_offset_42:16 is supported. See Table 52, “UMR Control Segment Layout,” on page 323.		No [Unlimited]
	6	null_mkey	If set, NULL_MKEY is reported in QUERY_SPECIAL_CONTEXT command, see Section 29.6.5, “QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers,” on page 1871.		Yes[Trust]
	5:0	log_max-klm_list_size	Log (base 2) of the maximum indirect KLM entries list (in MKey)		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
24h	~~23:5~~	resources_on_virtio_net_emulation_manager	If set, allocating resources on <u>VIRTIO_NET</u> emulation manager (instead of emulated device by a tunnel) is supported. This is done by setting ~~NVME~~ <u>VIRTIO_NET</u> _DEVICE_MULATION_OBJECT.resources_on_emulation_manager=1.		No [FW]
	~~5:22~~	resources_on_virtio_blk_emulation_manager	If set, allocating resources on <u>VIRTIO_BLK</u> emulation manager (instead of emulated device by a tunnel) is supported. This is done by setting ~~NVME~~ <u>VIRTIO_BLK</u> _DEVICE_MULATION_OBJECT.resources_on_emulation_manager=1.		No [FW]
	21:16	log_max_ra_req_dc	Log (base 2) of the maximum number of outstanding RDMA read/Atomic per DC QP as a requester		Yes [QUOTA]
	5:0	log_max_ra_res_dc	Log (base 2) of the maximum number of outstanding RDMA read/Atomic per DC QP as a responder		Yes [QUOTA]
28h	31	cc_roce_ecn_rp_classify_mode	When set RoCE ECN RP CC supports classifying flows for Congestion Control by either QP number or by destination IP.		
	30	cc_roce_ecn_rp_dynamic_rtt	When set RoCE ECN RP CC supports dynamic updates of dce_tcp_rtt according to the number of CC flows. Valid only for Mellanox 3rd generation NICs compatibility mode.		
	29	cc_roce_ecn_rp_dynamic_ai	When set RoCE ECN RP CC supports dynamic updates of rpg_ai_rate according to time elapsed since last CNP was received. Valid only for Mellanox 3rd generation NICs		
	28	cc_roce_ecn_rp_dynamic_g	When set RoCE ECN RP CC supports dynamic updates of dce_tcp_g according to the average number of CNPs received per QP per second. Valid only for Mellanox 3rd generation		
	27	cc_roce_ecn_rp_burst_decouple	When set RoCE ECN RP CC supports decoupling the transmission bursts of multiple CC QPs transmitting to the same destination IP. Valid only for Mellanox 3rd generation NICs compatibility mode.		

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
28h	25	release_all_pages	If set, the device releases all pages immediately for another function. SW can release all pages directly when getting release_all_pages == 1 in Table 234, “Pages Request Event Layout,” on page 502. Device reports 1 in max capability and 0 in current. SW that supports the flow should set 1 in the current capability.	RW	No [FW]
	24	sig_crc64_xp10	If set, CRC64_xp10 signature type is supported.		No [FW]
	23	sig_crc32c	If set, CRC32c signature type is supported.		No [FW]
28h	22	roce_accl	If set, ROCE_ACCL register will be supported. See Section 28.3.7, “ROCE_ACCL - ROCE Accelerator,” on page 1091.		No [FW]
	21:16	log_max_ra_req_qp	Log (base 2) of the maximum number of outstanding RDMA read/Atomic per QP as a requester	RW	Yes [QUOTA]
	5:0	log_max_ra_res_qp	Log (base 2) of the maximum number of outstanding RDMA read/Atomic per QP as a responder	RW	Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
2C	31	end_pad	End Padding in RX messages is supported. See Section 4.3.15, “End of Packet Padding (RX),” on page 205.		No [Unlimited]
	30	cc_query_allowed	Query Congestion Control parameters and statistics allowed. See Section 29.41, “Congestion Control Commands,” on page 2228		No [FW]
	29	cc_modify_allowed	Modify Congestion Control parameters and statistics allowed. See Section 29.41, “Congestion Control Commands,” on page 2228		No [FW]
	28	start_pad	Start Padding in RX messages is supported. See Section 4.3.14, “Start of Packet Padding (RX),” on page 205.		No [Unlimited]
	27	cache_line_128-byte	If set, 128 byte cache line size is supported. 0 means only 64 byte cache line is supported.	RW	No [Unlimited]
	26	gid_table_size_ro	If set, device will ignore the written value to gid_table_size.		No [Unlimited]
	25	pkey_table_size_ro	If set, device will ignore the written value to pkey_table_size.		No [Unlimited]
	24	rts2rts_qp_rmp	If set, modify xrq/rmp in QP context in RTS2RTS is supported. The modify of this field is best effort, and command will fail (BAD_RESOURCE_STATE) and no modifications done in the following cases: QP is in the middle of receiving a message or the original RMP is XRQ. During this command execution, incoming packets to this QP may be dropped, and no RNR_NACK will be sent for them.		No [Unlimited]
	23	rnr_nak_q_counters	If set, resp_rnr_nak and req_rnr_retries_exceeded Q counters are supported. See Section 29.35.3, “QUERY_Q_COUNTER – Query Q Counter,” on page 2188.		No [Unlimited]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
2Ch	22	rts2rts_qp_counters_set_id	If set, modify counters_set_id in QP context in RTS2RTS is supported.		No [Unlimited]
	21	rts2rts_qp_dscp	If set, modify dscp and f_dscp in QP context in RTS2RTS is supported.		No [Unlimited]
	20	gen3_cc_negotiation	If set, gen3_cc_negotiation_en in QUERY_CONG_STATUS and MODIFY_CONG_STATUS commands is supported.		
	19	vnic_env_int_rq_oob	If set, internal_rq_out_of_buffer counter is supported in QUERY_VNIC_ENV. See Section 29.34.3, on page 2181 [Internal] Should be reported only for eswitch manager.		No [FW]
	18	sbcam_reg	If set, SBCAM register is supported. See Section 28.21.23, “SBCAM - Shared Buffer Capabilities Mask Register,” on page 1490		No [FW]
	17	cwcamlreg	If set, CWCAM register is supported. See Section 28.20.17, “CWCAM - Congestion WRED Capabilities Mask Register,” on page 1447		No [FW]
2Ch	16	qcam_reg	If set, QCAM register is supported. See Table 785, “QCAM - QoS Capabilities Mask Register Layout,” on page 977.		No [FW]
	15:0	gid_table_size	GID table size. If gid_table_size_ro==1, the write is ignored. 0x0: gid_entries_8 0x1: gid_entries_16 0x2: gid_entries_32 0x3: gid_entries_64 0x4: gid_entries_128	RW	Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
30h	31	out_of_seq_cnt	If set, out of sequence counter in Q counters is supported. See Section 29.35, “Q Counters Commands,” on page 2185.		No [Unlimited]
	30	vport_counters	If set, Vport counters are supported. See Section 29.34, “Vport CountersMonitoring Commands,” on page 2176.		No [Unlimited]
	29	retransmission_q_counters	If set, retransmission Q counters is supported. See Section 29.35, “Q Counters Commands,” on page 2185.		No [Unlimited]
	28	debug	If set, debug capabilities are supported. Software should query debug capabilities for detailed capabilities. See Section 29.3.4.10, “Debug Capabilities,” on page 1777.		No [FW]
	27	modify_rq_counters_set_id	If set, modify counters_set_id in RQ context is supported. See Table 128, “CREATE_RQ and MODIFY_RQ Bit-mask,” on page 395		No [Unlimited]
	26	rq_delay_drop	If set, disabling packet drop due to lack of receive WQEs is supported.		No [FW]
	25:16	max_qp_cnt	Maximum number of QP counters sets.		Yes [QUOTA]
	15:0	pkey_table_size	PKey table size. If pkey_table_size_ro==1, the write is ignored. 0x0: entries_128 0x1: entries_256 0x2: entries_512 0x3: entries_1k 0x4: entries_2k 0x5: entries_4k	RW	Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
34h	31	vport_group_manager	Virtual port group manager. Responsible for enabling other Vports.		No [FW]
	30	vhca_group_manager	Virtual HCA group manager. Responsible for enabling other vHCAs in its group.		No [FW]
	29	ib_virt	If set, InfiniBand virtualization interface is supported. See Section 8.8, “HCA_Vport Context - HCA Virtual Port Context,” on page 295		No [FW]
	28	eth_virt	If set, Ethernet virtualization interface is supported. See Section 14.1, “Ethernet Virtualization,” on page 838		No [FW]
	27	vnuc_env_queue_counters	When set QUERY_VNIC_ENV supports current_error_queues, total_error_queues and send_queue_priority_update_flow counters. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap only for vport_group_manager.		Yes [Trust]
	26	ets	When set, ETS is supported. See Section 19, “Quality of Service,” on page 925.		No [FW]
	25	nic_flow_table	If set, NIC flow table mechanism is supported. To get detailed flow table capabilities, software should query NIC flow table cap. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.		No [Unlimited]
	24	eswitch_manager	If set, e-switch flow table mechanism is supported. To get detailed flow table capabilities, software should query Flow Table NIC Cap. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742.		No [FW]
	23	device_memory	If set, device internal memory capabilities is supported. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.		Yes [TRUST]
	22	mcam_reg	If set, MCAM register is supported. See Table 1546, “MCAM - Management Capabilities Mask Register Layout,” on page 1643.		No [FW]
	21	pcam_reg	If set, PCAM register is supported. See Table 952, “PCAM - Ports Capabilities Mask Register Layout,” on page 1146		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
34h	20:16	local_ca_ack_delay	The Local CA ACK Delay. This is the value recommended to be returned in Query HCA verb. The delay value in microseconds is computed using $4.096\mu s * 2^{(local\_ca\_ack\_delay)}$.		No [Unlimited]
	15	port_module_event	If set, Port Module Event is supported.		No [FW]
	14	enhanced_retransmission_q_counters	If set, enhanced retransmission Q counters is supported. See Section 29.35, "Q Counters Commands," on page 2185.		No [Unlimited]
	13	port_checks	If set, disabling port checks using PCMR register is supported.		No [FW]
	12	pulse_gen_control	When set, the device supports driver control over pulse generation through a GPIO pin.		No [FW]
	11	disable_link_up_by_init_hca	If set, device will not bring link up on INIT_HCA command. <i>[Internal] when keep_link_up in nvconfig ==1, FW bring device link_up when get power. when keep_link_up in nvconfig ==0, FW bring link up only on INIT_HCA. this field disable the option to bring link up on INIT_HCA.</i>	RW	No [FW]
	10	beacon_led	If set, Beacon LED is supported. See Section 28.26.24, "MLCR – Management LED Control Register," on page 1566.		No [FW]
	9:8	port_type	Indicates port type 0x0: IB 0x1: Ethernet		No [Unlimited]
	7:0	num_ports	Number of network ports.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
38h	31	reserved	Must Not Be Used		Yes [TRUST]
	30	pps	If set, query PPS parameters, using MTPPS register and PPS event mechanism is supported. See Section 13.3.1.1, “Auxiliary Pulse-Per-Second (PPS) Signals,” on page 833.		No [FW]
	29	pps_modify	If set, modifying PPS parameters using MTPPS register is supported. See Section 13.3.1.1, “Auxiliary Pulse-Per-Second (PPS) Signals,” on page 833.		No [FW]
38h	28:24	log_max_msg	Log (base 2) of the maximum message size in bytes supported by the device		No [Unlimited]
	23	multi_path_x-rc_rdma	If set, multi-path for RDMA operations (READ and WRITE) in XRC transport service is supported.		No [Unlimited]
	22	multi_path_d-c_rdma	If set, multi-path for RDMA operations (READ and WRITE) in DC transport service is supported.		No [Unlimited]
	21	multi_path_rc_rdma	If set, multi-path for RDMA operations (READ and WRITE) in RC transport service is supported.		No [Unlimited]
	20	traffic_fast_control	If set, INIT_SEGMENT.traffic_state is supported.		No [FW]
	19:16	max_tc	Number of Traffic Classes supported. The device supports TCs in range 0..max_tc-1. Value 0 indicates that 8 TCs are supported.		Yes [TRUST]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
38h	15	temp_warn_event	If set, Temperature Warning Event is supported. See Section 8.20.12.7, “Temp Warning Event,” on page 495.		No [FW]
	14	dcbx	If set, DCBX is supported. SW should query DCBX_PARAM register for more details. See Section 28.3.4, “DCBX_PARAM Register,” on page 1086.		No [FW]
	13	general_notification_event	If set, General Notification event is supported. See Section 8.20.12.8, “General Notification Event,” on page 495		No [FW]
	12	multi_prio_sq	If set, the device supports an indication from the SW that SQ is used for transmission of packets with multiple priorities. See Table 29, “NIC_Vport Context Layout,” on page 287.		Yes [TRUST]
	11	afu_owner	If set, AFU commands are supported. See Section 29.21, “AFU Command - Acceleration Functions Units commands,” on page 2070		TBD
	10	fpga	If set, FPGA is supported by the device. See TBD		No [FW]
	9	rol_s	If set, reset on secured magic packet is supported.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
38h	8	rol_g	If set, reset on magic packet is supported.		No [FW]
	7	ib_port_sniffer	If set, IB port sniffer is supported. See Section 29.38, "IB Port sniffer," on page 2217		No [FW]
	6	wol_s	If set, wake on secured magic packet is supported.		No [FW]
	5	wol_g	If set, wake on magic packet is supported.		No [FW]
	4	wol_a	If set, wake on ARP is supported.		No [FW]
	3	wol_b	If set, wake on broadcast packet is supported.		No [FW]
	2	wol_m	If set, wake on multicast packet is supported.		No [FW]
	1	wol_u	If set, wake on unicast packet is supported.		No [FW]
	0	wol_p	If set, wake on phy activity is supported.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
3Ch	31:16	stat_rate_support	Each bit corresponds to a value of Static Rate parameter in Address Vectors. For example, if bit 0 is set, then value of 0 (Full BW) is supported as Static Rate value.		No [Unlimited]
	15	sig_block_4048	If set, Block size of 4048 Bytes is supported for signature.		No [Unlimited]
	14	pci_sync_for_fw_update_event	If set, General notification event supports pci_sync_for_fw_update sub-event. [Internal] FW should report 1 in max and 0 in current. SW should set current to 1 in case max is 1. (This is an indication that driver is new and supported the flow).	RW	No [FW]
	13	init2rtr_drain_sigerr	If set, drain_sigerr in QP context can be modified in INIT2RTR command.		No [FW]
	12:8	log_max_extended_rnr_retry	Log(base 2) of the maximum value supported in QP.extended_rnr_retry. If set to 0, extended_rnr_retry_valid, extended_rnr_retry and extended_cur_rnr_retry are not supported. See Table 19, "QP Context Structure Layout," on page 258		
	7	init2_lag_tx_port_affinity	If set, lag_tx_port_affinity in QP context can be modified in INIT2INIT and INIT2RTR commands.		No [FW]
	6	flow_group_type_hash_split	When set, the device supports Flow Groups of type HASH_SPLIT, splitting traffic into buckets according to packet header hash calculation		No [FW]
	5	stat_rate_200	When this bit is set, rates of Gbps_56, Gbps_25, Gbps_100, Gbps_200, and Gbps_50 are supported in Address Path Structure Fields.stat_rate, Wqe.MLX Segment Format.static_rate & UD address vector field.stat_rate		No [FW]
	4	wqe_based_flow_table_update	If set, the device supports updating Flows in Flow Tables using a WQE. For detailed capabilities, See Table 1621, "WQE_BASED_FLOW_TABLE Capabilities Layout," on page 1799		Yes [Trust]
	3:0	cqe_version	CQE version. CQE format includes different format and fields depending on CQE version. See Table 145, "64B Completion Queue Entry Format Layout," on page 435.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
40h	31	compact_address_vector	WQE with compact Address Vector supported. See Table 109, “UD Address Vector Layout,” on page 362.		No [FW]
	30	eth_striding_wq	If set, Ethernet striding wq is supported.		No [Unlimited]
	29	Internal_keep-it_reserved			No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
40h	28	ipoib_enhanced_offloads	If set, IPoIB enhanced offloads is supported. To get detailed IPoIB Enhanced capabilities, SW should query IPoIB Enhanced capabilities using QUERY_HCA_CAP.op_mod = IPoIB enhanced offload capabilities.		No [Unlimited]
	27	ipoib_basic_offloads	If set, IPoIB basic offloads are supported. To get detailed IPoIB basic capabilities, SW should query IPoIB basic capabilities using QUERY_HCA_CAP.op_mod = IPoIB basic offload capabilities.		No [Unlimited]
	26	ib_link_list_striding_wq	If set, IBstriding WQ for link list WQ is supported.		No [Unlimited]
	25	repeated_block_disabled	When set, repeated/strided block MKey is not supported.		Yes [Trust]
	24	umr_modify_entity_size_disabled	When set, UMR may not modify page/buffer size (log_entity_size) of an MKey		No [FW]
	23	umr_modify_atomic_disabled	When set, UMR may not modify atomic privileges (a) of an MKey		No [FW]
	22	umr_indirect_mkey_disabled	When set, Indirect MKeys cannot be set to be UMR-able		No [FW]
	21:20	umr_fence	The required fence type for UMR WQEs. 0x0: STRONG - a strong fence is required. (Internal CX4) 0x1: SMALL - a small fence is required. (Internal CX5) 0x2: NONE - No fence is required. (Internal - future)		No [Unlimited]
	19	dc_req_sctr_data_cqe	If set, scatter data to CQE for requester in DC is supported.		No [Unlimited]
	18	dc_connect_qp	If set, DC connect QP is supported.		No [Unlimited]
	17	dc_cnak_trace	If set, dc cnak trace is supported. See Section 16.2.4, "DC CNAK Trace," on page 874		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
40h	16	drain_sigerr	If set, QP moves to SQ draining upon signature hand-over error. See Section 17.3.14.2, “Signature Handover Completion Report,” on page 895.		No [Unlimited]
	15:14	cmdif_checksum	0x0: disabled - command interface signature is neither checked for input nor for output 0x1: initial_state - command interface signature is not checked on input, but is generated by device for output 0x3: enabled - command interface signature is both checked for input and generated for output. [EMC] when “e” bit is set, setting state to 0x1 will behave the same as 0x3 (enabled)	RW	No [FW]
	13	sigerr_cqe	Signature error CQE supported.		No [Unlimited]
	12	e_psv	Extended psv support. When enabled, the device will collect signature errors extended data and will report them in QUERY_PSV.	RW	No [FW]
	11	wq_signature	WQE signature check on WQ - (for SQ, RQ and QP).		No [Unlimited]
	10	sctr_data_cqe	Scatter Data to CQE is supported.		No [Unlimited]
	9	bsf_in_create_mkey	If set, SW can set a list of BSFs in CREATE_MKEY command.		Yes [TRUST]
	8	sho	Signature handover is supported.		Yes [TRUST]
	7	tph	tph - tlp processing hints are supported.		TBD

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
40h	6	rf	If set, the HCA support remote fence for RDMA read.		No [FW]
	5	dct	Dynamically Connected transport mode is supported.	RW	Yes [TRUST]
	4	qos	If set, quality of service capabilities are supported. To get detailed quality of service capabilities, software should query quality of service cap. See Section 29.3.4.9, “Quality of Service Capabilities,” on page 1773.		Yes [TRUST]
	3	eth_net_offloads	If set, Ethernet networking offloads are supported. To get detailed Ethernet capabilities, software should query Ethernet cap. See Section 29.3.4.5, “Networking Offload Capabilities,” on page 1734.		No [Unlimited]
	2	roce	If set, RoCE operations are supported. To get detailed RoCE capabilities, software should query RoCE cap. See Section 29.3.4.4, “RoCE Capabilities,” on page 1732.		Yes [TRUST]
	1	atomic	If set, atomic operations are supported. To get detailed atomic capabilities, software should query atomic cap See Section 29.3.4.3, “Atomic Capabilities,” on page 1728.		Yes [TRUST]
	0	extended_retry_count	If set, extended_retry_count_valid, extended_retry_count and extended_cur_retry_count in QP context are supported. See Table 19, “QP Context Structure Layout,” on page 258		Yes [TRUST]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
44h	31	cq_oi	If set, cq.oi can be modified by MODIFY_CQ command. See Section 29.9.4, “MODIFY_CQ – Modify CQ Parameters,” on page 1894.		No [Unlimited]
	30	cq_resize	If set, resizing CQ is enabled. See Section 8.19.6, “Resizing a CQ,” on page 453		No [Unlimited]
	29	cq_moderation	If set, CQ moderation is enabled by MODIFY_CQ Command. See Table 1872, “MODIFY_CQ Input Structure Layout,” on page 1894		Yes [TRUST]
	28	cq_period_mode_modify	If set, CQ.cq_period_mode can be modified by MODIFY_CQ. See Section 29.9.4, “MODIFY_CQ – Modify CQ Parameters,” on page 1894		Yes [TRUST]
	27	cq_invalidate	If set, setting CQ in invalid state using MODIFY_CQ command is supported. See Section 29.9.4, “MODIFY_CQ – Modify CQ Parameters,” on page 1894		No [Unlimited]
	25	cq_eq_remap	If set, cq.eqn can be modified by MODIFY_CQ. i.e. CQ to EQ remapping is supported. See Section 29.9.4, “MODIFY_CQ – Modify CQ Parameters,” on page 1894.		No [Unlimited]
	24	pg	If set, on-demand paging is supported. For more details, See Section 9.1.2, “On-demand Paging Support,” on page 626.		Yes [TRUST]
	23	block_lb_mc	Block loopback multicast supported. Can be enabled per QP using QP.block_lb_mc_en.		No [Unlimited]
	22	exponential_back-off	When set, exponentially increasing timeout value is supported. See Section 13.2.5, “Exponential Backoff Timer,” on page 832.		Yes [TRUST]
	21	scqe_break_moderation	Completion Event Moderation breakage by solicited CQE feature is supported.		Yes [TRUST]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
44h	20	cq_period_start_from_cqe	If set, cq_period_mode =1 is supported.(i.e. cq_period timer restarts upon completion generation). See Table 171, "Completion Queue Context Layout," on page 467.		Yes [TRUST]
	19	cd	If set, core direct is supported		Yes [TRUST]
	18	atm	This field replaced with a new field called 'atomic'		No [FW]
	17	apm	If set, Automatic Path Migration is supported		Yes [TRUST]
	16	vector_calc	If set, vector calc operations is supported. To get detailed vector calc capabilities, software should query vector calc cap. See Section 29.3.4.8, "Vector Calc Capabilities," on page 1772.		Yes [TRUST]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
44h	15	umr_ptr_rlkey	If set, the reserved lkey is allowed to be used in a UMR WQE data segment of type UMR pointer as described in Table 54, “UMR Pointer Description Argument Format,” on page 326 .		No [Unlimited]
	14	imaicl	Internal mkey access is considered local. If set, internal mkey access (access to an mkey via a pointer of an indirect mkey) will always be considered a local access, thus remote access flags will not be checked on the internally accessed mkey. Note: A given mkey may be accessed internally by one packet and externally by another packet. See Section 18.3.1, “User-Mode Memory Registration (UMR),” on page 918 .		No [Unlimited]
	13	qp_packet_based	if set, packet based e2e credit mode is supported in req_e2e_credit_mode in QP.		No [FW]
	12	ib_cyclic_striding_wq	If set, striding CYCLIC WQ is supported for IB traffic where QPs are attached to an XRQ with a WQ with wq_type == 0x2 or 0x3.		No [FW]
	11	ipoib_enhanced_pkey_change	When set, PKey change does not require recreating the underlay IPoIB QPs.		No [FW]
	10	initiator_src_dct_in_cqe	If set, CQ supports initiator_src_dct. [Internal] In ConnectX5 must be reported only in EMC mode as its global configuration.		No [FW]
	9	qkv	If set, QKey Violation Counter is supported		No [Unlimited]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
44h	8	pkv	If set, PKey Violation Counter is supported		No [Unlimited]
	7	set_deth_sqpn	If set, SW can control the DETH source QP number on UD QPs with the QP.deth_sqpn. See Table 19, “QP Context Structure Layout,” on page 258.		No [Unlimited]
	6	rts2rts_primary_sl	If set, RTS2RTS command supports setting sl in primary path. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.		No [Unlimited]
	5	initiator_src_dct	If set, initiator_src_dct in QP Context is supported. See Table 19, “QP Context Structure Layout,” on page 258		No [Unlimited]
	4	dc_v2	If set, DC transport service version 2 is supported.		Yes [TRUST]
	3	xrc	If set, XRC transport is supported		Yes [TRUST]
	2	ud	If set, UD Transport is supported		Yes [TRUST]
	1	uc	If set, UC Transport is supported		Yes [TRUST]
	0	rc	If set, RC Transport is supported		Yes [TRUST]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
48h	31	uar_4k	If set, UARs are of size 4KB. In this mode, ALLOC_UAR allocates a group of UARs of size HCA_CAP.number_of_uars_per_page. [Internal] FW should set max to 1 and current to 0 to save compatibility.	RW	No [Unlimited]
	21:16	uar_sz	UAR Area Size = 1MB * 2 ^{uar_sz}		Yes [QUOTA]
	12:8	log_max_dc_cnak_qps	Log (base 2) of the maximum number of supported DC_CNAK QPs. The device distributes the connects randomly between the opened QPs.		Yes [QUOTA]
	7:0	log_pg_sz	Log (base 2) of the minimum system page size supported. For proper operation it must be less than or equal to the minimum page size of the hosting platform (CPU).		No [FW]
4Ch	31	bf	If set to '1' then BlueFlame may be used		TBD
	30	driver_version	If set, SET_DRIVER_VERSION command is supported and should be used by the driver which performs start-up. See Section 29.3.19, "SET_DRIVER_VERSION," on page 1829.		No [Unlimited]
	29	pad_tx_eth_packet	If set, device automatically pads Ethernet packets shorter than 64 bytes to 64 bytes.		No [Unlimited]
	28	query_driver_version	If set, the device supports querying the driver version that was set by SET_DRIVER_VERSION, using QUERY_DRIVER_VERSION. See Section 29.3.20, "QUERY_DRIVER_VERSION," on page 1830		No [FW]
	27	max_qp_retry_freq	If set, QP maximum retry frequency limit is supported. See configuration of max_qp_retry_limit and max_qp_retry_period in Nic Vport Context.		No [FW]
	26	qp_by_name	If set, SW can choose QP number in CREATE_QP command.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
4Ch	25	mkey_by_name	If set, SW can choose Mkey number in CREATE_MKEY command. [Internal] - FW must report the max capability according to mkey_by_name in NV_SW_OFFLOAD_CONFIG, and 0 in current capability. (need to make sure default of NVCONFIG is also 0). In case max capability of this field is 1, SW must set the current capability to 1, otherwise FW must fail driver initialization. Note - Need to make sure SW queries terminate_scatter_list_mkey and repeated_mkey from QUERY_SPECIAL_CONTEXT beside this feature.	RW	Yes
	23	suspend_qp_uc	If set, Suspend/Resume is supported for UC QPs. See TBD		No [FW]
	22	suspend_qp_ud	If set, Suspend/Resume is supported for UD QPs. See TBD		No [FW]
	21	suspend_qp_rc	If set, Suspend/Resume is supported for RC QPs. See TBD		No [FW]
	20:16	log_bf_reg_size	Log (base 2) of BlueFlame max register size in bytes.		Yes [QUOTA]
	9:8	lag_dct	Indicates mode of DCT LAG support. 0x0: NOT_SUPPORTED - DCT is not supported following a CREATE_LAG command 0x1: SW_ASSISTED - SW assisted LAG DCT is supported (see <cross ref to the chapter that explains> for details) 0x2: HW_ONLY - DCT with LAG is supported without SW assistance.		No [FW]
	7	lag_tx_port_affinity	If set, lag_tx_port_affinity in QP and TIS is supported. even if LAG isn't created by CREATE_LAG. Note that this can be used only if num_lag_ports>1.		Yes [FW]
	6	lag_native_fdb_selection	If set, NATIVE_FDB mode (mode 1) will be supported in CREATE_LAG command.		No [FW]
	5	must_be_0			

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
4Ch	4	lag_master	If set, SW is allowed to create and manage LAG. Valid only when HCA_CAP.num_lag_ports>1. See Chapter 14, “Virtualization” on page 838.		No [FW]
	3:0	num_lag_ports	The number of lag ports supported. If greater than 1, LAG features are supported and indicates the number of lag ports supported. May likely be greater than HCA_CAP.num_ports. See Chapter 14, “Virtualization” on page 838.		No [FW]
50h	31:16	num_of_diagnostic_counters	Indicates number of diagnostic counters supported by device. Software should query debug capabilities for detailed capabilities. See Section 29.3.4.10, “Debug Capabilities,” on page 1777		No [FW]
50h	15:0	max_wqe_sz_sq	Max descriptor size in bytes for the SQ. SQ WQEs must not be bigger than indicated here. RDMA-read WQE size cannot exceed 512 bytes.		No [FW]
54h	15:0	max_wqe_sz_rq	Max descriptor size in bytes for the RQ and SRQ. RQ and SRQ WQEs must not be bigger than indicated here.		No [FW]
58h	31:16	max_flow_counter_31_16	MSB of the maximum number of flow counters.		Yes [QUOTA]
	15:0	max_wqe_sz_sq_dc	Max descriptor size in bytes for the SQ of DC type. SQ WQEs must not be bigger than indicated here.		No [FW]
5Ch	24:0	max_qp_mcg	Maximum number of QPs per multicast group.		Yes [QUOTA]
60h	31:16	mlnx_tag_ether-type	Ethernet type value which indicates Mellanox tagging. 0 means not supported. See Section 4.7, “MLNX ETHERTYPE TAGGING,” on page 214.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
60h	15:8	flow_counter_bulk_alloc	Bitmask field indicates the supported bulk sizes in ALLOC_FLOW_COUNTER command. See Table 2174, “ALLOC_FLOW_COUNTER Input Structure Layout,” on page 2038. Bit 0: BULK_128 - allocate 128 counters. Bit 1: BULK_256 - allocate 256 counters. Bit 2: BULK_512 - allocate 512 counters. Bit 3: BULK_1K - allocate 1K counters. Bit 4: BULK_2K - allocate 2K counters. Bit 5: BULK_4K - allocate 4K counters. Bit 6: BULK_8K - allocate 8K counters. Bit 7: BULK_16K - allocate 16K counters.		No [FW]
	7:0	log_max_mcg	Log (base 2) of the maximum number of multicast groups.		Yes [QUOTA]
64h	28:24	log_max_transport_domain	Log (base 2) of the maximum number of Transport Domains.		Yes [QUOTA]
	20:16	log_max_pd	Log (base 2) of the maximum number of PDs.		Yes [QUOTA]
	4:0	log_max_xrxd	Log (base 2) of the maximum number of XRCDs.		Yes [QUOTA]
68h	31	nic_receive_steering_discard	When set QUERY_VNIC_ENV supports nic_receive_steering_discard counter. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap to all functions(ECPF/PF/VF).		No [Unlimited]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
68h	30	receive_discard_vport_down	When set QUERY_VNIC_ENV supports receive_discard_vport_down counter. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap to all functions(ECPF/PF/VF).		No [Unlimited]
	29	transmit_discard_vport_down	When set QUERY_VNIC_ENV supports transmit_discard_vport_down counter. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap to all functions(ECPF/PF/VF).		No [Unlimited]
	28	eq_overrun_count	When set, the device supports Completion event and Async event EQ overrun counter for a vNIC in QUERY_VNIC_ENV. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap only for vport_group_manager.		No [Unlimited]
	27	nic_receive_steering_depth	when set, the device support reporting of a the steering depth of the NIC Receive Flow Table in QUERY_VNIC_ENV. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap only for resource_manager.		No [Unlimited]
	26	invalid_command_count	When set, the device supports Invalid Command counter for a vNIC in QUERY_VNIC_ENV. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap to all functions(ECPF/PF/VF).		No [Unlimited]
	25	quota_exceeded_count	When set, the device supports Quota Exceeded counter for a vNIC in QUERY_VNIC_ENV. See Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181 [Internal] Device should report this cap to all functions(ECPF/PF/VF).		No [Unlimited]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
68h	24	must_not_use	[Internal] Must not be used. deleted cap from FW		
	23:16	log_max_flow_counter_bulk	Log (base 2) of the maximal number of flow counters that can be queried by a single QUERY_FLOW_COUNTER command. See Table 2182, “QUERY_FLOW_COUNTER Input Structure Layout,” on page 2041.		No [FW]
	15:0	max_flow_counter_15_0	LSB of the maximum number of flow counters. [Internal] FW must not report this field as zero and must report the first default value due to compatibility issues.		Yes [QUOTA]
6Ch	31	modify_tis	If set, MODIFY_TIS command is supported. See Section 29.13.2, “MODIFY_TIS – Modify TIS,” on page 1956.		No [Unlimited]
	30	flow_counters_dump	If set, device supports dump of flow counters to memory by QUERY_FLOW_COUNTER command instead of returning counters in the command’s outbox.		
	28:24	log_max_rq	Log (base2) of the number of RQ supported. 0 - feature not supported.		Yes [QUOTA]
	20:16	log_max_sq	Log (base2) of the number of SQ supported. 0 - feature not supported.	RW	Yes [QUOTA]
	12:8	log_max_tir	Log (base2) of the number of TIR supported. 0 - feature not supported.		Yes [QUOTA]
	4:0	log_max_tis	Log (base2) of the number of TIS supported. 0 - feature not supported.		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
70h	31	basic_cyclic_rcv_wqe	0: cyclic receive WQE always includes the 16 byte of ctrl (signature field). 1: cyclic receive WQE includes the 16 bytes of ctrl only when RMP.basic_cyclic_rcv_wqe/XRC_SRQ.basic_cyclic_rcv_wqe is enabled. [Internal] - 0 in Golan and 1 in shomron.		No [Unlimited]
	29	mem_rq_rmp	If set, mem_rq_type of type MEMORY_RQ_RMP in RQ context is supported. See Table 125, "RQ Context Fields," on page 386		No [FW]
	28:24	log_max_rmp	Log (base2) of the number of RMPs supported. 0 - feature not supported.		Yes [QUOTA]
	20:16	log_max_rqt	Log (base2) of the number of RQTs supported. 0 - feature not supported.		Yes [QUOTA]
	12:8	log_max_rqt_size	Log (base2) of max RQT size.		No [FW]
	4:0	log_max_tis_per_sq	Log (base2) of the number of TIS supported per SQ.		No [FW]
74h	31	ext_stride_num_range	When set, the device supports the following values for the number of strides per WQE of a Striding WQ: 8, 16, 32..., 256		No [FW]
	28:24	log_max_stride_sz_rq	Log (base2) of the maximum size (in bytes) of RQ stride.		No [FW]
	20:16	log_min_stride_sz_rq	Log (base2) of the minimum size (in bytes) of RQ stride.		No [FW]
	12:8	log_max_stride_sz_sq	Log (base2) of the maximum size (in bytes) of SQ stride.		No [FW]
	4:0	log_min_stride_sz_sq	Log (base2) of the minimum size (in bytes) of SQ stride.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
78h	31	hairpin	If set, hairpin is supported. See Section 4.8, "Hairpin," on page 215		Yes [TRUST]
	28:24	log_max_hairpin_queues	Log (base 2) of the maximum hairpin SQs and RQs supported.		Yes [QUOTA]
	23	hairpin_ib_raw	If set, hairpin for IB native traffic is supported. See Section 4.8, "Hairpin," on page 215		Yes [TRUST]
	22	hairpin_eth2ipoib	If set, hairpin for ETH traffic to IPoIB traffic is supported. See Section 4.8, "Hairpin," on page 215		Yes [TRUST]
	21	hairpin_ipoib2eth	If set, hairpin for IPoIB traffic to ETH traffic is supported. See Section 4.8, "Hairpin," on page 215		Yes [TRUST]
	20:16	log_max_hairpin_wq_data_sz	Log (base 2) of the maximum size of data buffer on hairpin WQ in granularity of bytes.		Yes [QUOTA]
	12:8	log_max_hairpin_num_packets	Log (base 2) of the maximum number of packets allowed on hairpin WQ.		Yes [QUOTA]
78h	4:0	log_max_wq_sz	Log (base 2) of the maximum number of WQEs allowed on the WQ.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
7Ch	31	nic_vport_change_event	If set, Nic vport change event mechanism supported. See Section 14.1.6, “NIC Vport Change Notification,” on page 844.		No [FW]
	30	disable_local_lb_uc	If set, disabling local loopback for unicast is supported in nic vport context using disable_uc_local_lb. See Table 29, “NIC_Vport Context Layout,” on page 287.		No [Unlimited]
	29	disable_local_lb_mc	If set, disabling local loopback for multicast is supported in nic vport context using disable_mc_local_lb. See Table 29, “NIC_Vport Context Layout,” on page 287.		No [Unlimited]
	28:24	log_min_hairpin_wq_data_sz	Log (base 2) of the minimum size of data buffer on hairpin WQ in granularity of bytes.		No [FW]
	23	system_image_guid_modifiable	If set, modifying system_image_guid is supported in NIC vport context. See Table 29, “NIC_Vport Context Layout,” on page 287.		No [FW]
	20:16	log_max_vlan_list	Log (base2) of the maximum size of VLAN list used in nic_vport_context. See Table 29, “NIC_Vport Context Layout,” on page 287		Yes [QUOTA]
	12:8	log_max_current_mc_list	Log (base2) of the maximum size of current_mc_mac_address list used in nic_vport_context. See Table 29, “NIC_Vport Context Layout,” on page 287.		Yes [QUOTA]
	4:0	log_max_current_uc_list	Log (base2) of the maximum size of current_uc_mac_address list used in nic_vport_context. See Table 29, “NIC_Vport Context Layout,” on page 287.		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
80h-84h	64	general_obj_types	Bit mask indicates which general objects are supported. See Section 29.23, “General Object Commands,” on page 2079. Bit 0x6: NVME_DEVICE_EMULATION_OBJECT Bit 0x7: NVME_SQ Bit 0x8: SW_ICM Bit 0x9: NVME_CQ Bit 0xa: FLOW_METER Bit 0xb: GENEVE_TLV_OPTION Bit 0xc: DEK Bit 0xd: VIRTQ_NET_Q - Detailed capabilities are reported in VIRTIO Emulation capabilities. See Table 1615, “VIRTIO Emulation Capabilities Layout,” on page 1793 Bit 0xe: VHCA_TUNNEL_OBJECT Bit 0xf: NVME_NAMESPACE Bit 0x10: NVME_CTRL_COUNTERS Bit 0x11: NVME_NAMESPACE_COUNTERS Bit 0x12: NVME_SQ_COUNTERS_COUNTERS Bit 0x13: IPSEC_OFFLOAD Bit 0x14: DEVICE_OBJECT Bit 0x15: VIRTIO_NET_DEVICE_EMULATION Bit 0x16: VIRTIO_BLK_DEVICE_EMULATION Bit 0x17: VIRTQ_BLK_Q Bit 0x18: MATCH_DEFINER Bit 0x19: NVMF_BACKEND_CONTROLLER Bit 0x1A: NVMF_NAMESPACE Bit 0x1B: NVMF_NAMESPACE_COUNTER_SET Bit 0x1C: VIRTIO_Q_COUNTERS Bit 0x1D: IMPORT_KEY Bit 0x1E: CREDENTIAL Bit 0x1F: CRYPTO_LOGIN Bit 0x20: FLOW_SAMPLER Bit 0xff00: SF - Note that this event and the following events can’t be created by general object command. these allocation is a placeholder for numbers that can be used in future affiliated events [Internal] FW must report them 0. Bit 0xff01: MKEY Bit 0xff02: QP Bit 0xff03: PSV Bit 0xff04: RMP Bit 0xff05: XRC_SRQ Bit 0xff06: RQ Bit 0xff07: SQ Bit 0xff08: TIR Bit 0xff09: TIS Bit 0xff0a: DCT		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
88h	27:24	steering_format_version	Indicates the device steering logic format. If the device driver does not recognize the current version, SW owned tables should not be used. Valid only for Tables with Sw_owner_v2 set.[Internal] 0x0 – CONNECTX_5 0x1 – CONNECTX_6DX		No [SW]
	23:0	create_qp_start_hint	Indicates a hint to the device which QP number is recommended to start creating QPs. Supported only if !=0 and 0 means use device creation hint. Note that the reported and requested value is a best effort hint for creating a QP. [Internal] FW need to report value !=0 (1 is good) in max cap and current hint default in current cap.		No [FW]
8Ch	31	must_be_0	[Internal] Was TLS and deprecated. FW must report 0. If set, TLS offload is supported. Detailed capabilities are reported in See Table 1609, “TLS Capabilities Layout,” on page 1789		

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
8Ch	30	ats	If set, the device supports ATS and it has been enabled in the PCIe extended capability. Only when it is set, SW may set Mkey.ma_translation_mode==ATS and UMEM.ats==1.		Yes
	28:24	log_max_uctx	Log(base 2) of maximum supported UCTX objects. 0 means not supported.		
	23	aes_xts			Yes [Trusted]
	22	crypto			Yes [Trusted]
	21	ipsec_offload	When set IPsec Offload capabilities are supported. Detailed capabilities are reported in Table 1611, "IPSEC Capabilities Layout," on page 1790		Yes [Trusted]
	20:16	log_max_umem	Log(base 2) of maximum supported UMEM objects. 0 means not supported.		
	15:0	max_num_eqs	Maximum number of EQs supported by the device. Valid only when not 0. In case it's 0, SW should get number of EQs from HCA_CAP.num_of_eqs.		Yes [QUOTA]
90h	30	tls_tx	If set, TLS offload For transmit traffic is supported. Detailed capabilities are reported in Table 1609, "TLS Capabilities Layout," on page 1789		No [FW]
	29	tls_rx	If set, TLS offload For received traffic is supported. Detailed capabilities are reported in Table 1609, "TLS Capabilities Layout," on page 1789		No [FW]
	28:24	log_max_l2_table	Log (base2) of the maximum size of L2 Table. See Section 29.19, "L2 Table Commands," on page 2058.		No [FW]
	15:0	log_uar_page_sz	Log (base 2) of UAR page in 4Kbyte chunks.	RW	No [Unlimited]
94h	31	e	[EMC] EMC mode. EMC-only features and APIs are enabled	RW	No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
98h	31:0	device_frequency_mhz	Internal device frequency given in MHz. Valid only if non-zero. This field is deprecated and replaced by device_frequency_khz.		No [Unlimited]
9Ch	31:0	device_frequency_khz	Internal device frequency given in KHz. Valid only if non-zero. When supported, this field reports more accurate value of the device frequency comparing to device_frequency_khz. See Section 8.19.12, "CQE Timestamping," on page 471		No [Unlimited]
A0h	31	capi	If set, CAPI is supported. See Section 22, "CAPI- Coherent Accelerator Programming Interface," on page 1035		No [FW]
	30	create_pec	If set, CREATE_CAPI_PEC is supported. See Section 22, "CAPI- Coherent Accelerator Programming Interface," on page 1035		No [FW]
A0h	29	nvmf_target_offload	If set, NVMe-oF target offload is supported. See Section 8.21.3, "NVM Express over Fabric (NVMe-oF) Target Offload Context," on page 519.		No [FW]
A0h	28	capi_invalidate	If set, CAPI invalidation by INIT_SEGMENT.capi_invalidate is supported.		No [FW]
	4:0	log_max_pasid	Log (base 2) of the maximum PASIDs supported. Section 22, "CAPI- Coherent Accelerator Programming Interface," on page 1035.		No [FW]
A4h	31:0	num_of_uars_per_page	Number of 4KB UARs per system page. Valid only when uar_4kb==1.		No [Unlimited]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
A8h	31:0	flex_parser_protocols	Bit mask indicates which Flex parser protocols are already enabled. The enable is done by NV_FLEX_PARSER_CONF Tlv. Bit 0: flex_ipv4_over_vxlan_enabled Bit 1: flex_ipv4_over_ip_enabled Bit 2: flex_ipv6_over_ip_enabled Bit 3: flex_geneve_enabled Bit 4: flex_cw_mpls_over_gre_enabled Bit 5: flex_cw_mpls_over_udp_enabled Bit 6: internal_not_use Bit 7: flex_vxlan_gpe_enabled Bit 8: flex_icmp_enabled Bit 9: flex_icmpv6_enabled Bit 10: flex_geneve_tlv_option_0_enabled Bit 11: flex_gtp_u_enabled		
ACh	31:24	max_geneve_tlv_options	Maximum number of supported geneve TLV options that can be matched		
	20:16	max_geneve_tlv_option_data_len	The maximum value that can be set in data_length field in GENEVE_TLV_OPTION general object. See		
ACh	12:8	log_max_guaranteed_connections	Log (base2) of the maximum number of the total guaranteed outstanding connections.		Yes [QUOTA]
	4:0	log_max_dct_connections	Log (base2) of maximum number of outstanding connections that can be assigned per DCT.		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
B0h	31:24	log_max_atomic_size_qp	Log (base 2) of the maximum atomic operand size in bytes for all qps. Removed.	RW	Yes [QUOTA]
	20:16	log_max_dci_stream_channels	Log (base 2) of the maximum DCI stream channels supported per DCI.		Yes [QUOTA]
	12:8	log_max_dci_error_streams	Log (base 2) of the maximum DCI error stream channels supported per DCI before a DCI move to error state.		Yes [QUOTA]
	7:0	log_max_atomic_size_dc	Log (base 2) of the maximum atomic operand size in bytes for DC qps Removed		Yes [QUOTA]
B4h	31:16	max_multi_user_group_size	Maximum size of slave QPs that can be attached to a multi user master QP. 0 means feature not supported See Section 8.3.4.5, "Multi User QP," on page 256		No [FW]
	12:8	log_max_dek	Log(base 2) of maximum DEK Objects that are supported. 0 means not supported.		Yes [Quota]
B4h	3	mini_cqe_resp_stride_index	If set, Resp_ByteCnt_Checksum_Stride_Index format in mini_cqe_res_format is supported. See Table 172, "Completion Queue Context Field Descriptions," on page 468		No [FW]
	2	cqe_128_always	If set, CQE of size byte always is supported. See cqe_sz field in CQ Context. See Table 171, "Completion Queue Context Layout," on page 467		No [FW]
	1	cqe_compression_128b	If set, CQE compression for 128 Byte CQE is supported. See Section 8.19.9, "CQE Compressing," on page 459		No [FW]
	0	cqe_compression	CQE Compression is supported, For details See Section 8.19.9, "CQE Compressing," on page 459		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
B8h	31:16	cqe_compression_timeout	CQE Compression session expiration timeout in usecs. Valid only if cqe_compression = 1. <i>For details See Section 8.19.9, "CQE Compressing," on page 459</i>		No [FW]
	15:0	cqe_compression_max_num	Max number of compressed and aggregated CQEs in a single session. Valid only if cqe_compression = 1. <i>For details See Section 8.19.9, "CQE Compressing," on page 459</i>		No [FW]
BCh	28:24	wqe_based_flow_table_update_dest_type_offset	Indicates the number of bits dedicated for destination_id, and the bit offset at which the destination_type starts when constructing destination_handle in Flow Update Segment. Valid only when HCA_CAP.wqe_based_flow_table_update==1.		No [FW]
BCh	19:16	log_max_tm_offloaded_op_size	Log (base 2) of the maximal IB payload size of offloaded tagmatching message in granularity of bytes (includes tag matching header).		Yes [QUOTA]
	15	tag_matching	When set, Tag Matching is supported. See Section 20.3, "Tag Matching and Rendezvous Offload," on page 986		Yes [TRUST]
	14	rndv_offload_rc	When set, rendezvous offload for RC transport service is supported. See Section 20.3, "Tag Matching and Rendezvous Offload," on page 986		Yes [TRUST]
	13	rndv_offload_dc	When set, rendezvous offload for DC transport service is supported. See Section 20.3, "Tag Matching and Rendezvous Offload," on page 986		Yes [TRUST]
	12:8	log_tag_matching_list_sz	Log (Base 2) of the maximum number of tags SW is allowed to append for tag matching topology.		Yes [QUOTA]
	7	modify_xrq	If set, MODIFY_XRQ command will be supported. See Section 29.27.4, "MODIFY_XRQ – Modify XRQ," on page 2120		No [FW]
	4:0	log_max_xrq	Log (Base 2) of the maximum number of XRQs supported.		Yes [QUOTA]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
C0h	31:24	affiliate_nic_vport_criteria	A bitmask that indicated which vport affiliation criteria are supported in NIC_VPORT_CONTEXT. See Section 8.7, “NIC_Vport Context - NIC Virtual Port Context,” on page 287 Bit 0: RDMA_AFFILIATION – if set, RDMA affiliation is supported.		No [Unlimited]
	23:16	native_port_num	The device physical port number that the VHCA is natively associated with. 0 means field is invalid. See Section 14.4, “Multi-port vHCA,” on page 845		No [FW]
	15:8	num_vhca_ports	The number of supported VHCA ports. 0 means the VHCA supports a single port (0 and 1 are equivalent). Applicable only when HCA_CAP.num_ports=1 (reserved otherwise). See Section 14.4, “Multi-port vHCA,” on page 845		No [FW]
	2	trusted_vnic_vhca	When set, the vNIC/vHCA is as trusted as vhca_group_manager.		Yes [trust]
	1	sw_owner_id	If set, the device will accept sw_owner_id by INIT_HCA Command. See Section 14.4, “Multi-port vHCA,” on page 845		No [FW]
	0	reserve_not_to_use			No [FW]
C4h	31:16	max_num_of_monitor_counters	The maximum number of counters that can be monitored by SET_MONITOR_COUNTERS command.		Yes [trust]
	15:0	num_ppent_monitor_counters	The number of ppent group counters that are supported in PPCNT group by SET_MONITOR_COUNTERS command.		No [FW]
C8h	15:0	num_q_monitor_counters	The number of Q group counters that are supported in Q Counters by SET_MONITOR_COUNTERS command.		Yes [trust]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
D0h	31	sf	If set, Sub-Functions is supported (See Section 14.3, “Sub-Function,” on page 845). In case max capability is 1, SW should set current capability to 1 too. [Internal] FW must report: Current CAP Always 0. Max CAP 1 only for eswitch manager. If eswitch manager sets it's current to 1, then report to othernot eswitch manager PF 1 in max CAP.	RW	No [FW]
	30	sf_set_partition	If set, modify of sub-function partitions configuration is supported by SET_SF_PARTITIONS command. otherwise, SW can only read the configuration by QUERY_SF_PARTITIONS command. Valid only when sf==1.		No [FW]
	28:24	log_max_sf	Log(base 2) of the maximum supported Sub-Functions. Valid only when sf==1.		No [FW]
	15:8	log_min_sf_size	Log(base 2) of the minimum bar size of sub-function on in granularity of 4KB. Valid only when sf==1.		No [FW]
	7:0	max_num_sf_partitions	Maxim number of sub-function partitions that can be created by SET_SF_PARTITIONS command. Valid only when sf==1.		No [FW]
D4	31:0	uctx_permission	Bit mask field indicates which capabilities are supported in cap in UCT Context.		Yes [trust]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
D8h	27:24	flex_parser_id - geneve_tlv_option_0	Indicate the Flex Parser ID used to parse geneve_tlv_option_0 header. Valid only when geneve_tlv_option_0_enabled in HCA_CAP.flex_parser_protocols is set.		
	23:20	flex_parser_id_icmp_dw1	Indicates the Flex Parser ID used to parse DW 1 of ICMP header. Valid only when flex_icmp_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
	19:16	flex_parser_id_icmp_dw0	Indicates the Flex Parser ID used to parse DW 0 of ICMP header. Valid only when flex_icmp_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
	15:12	flex_parser_id_icmpv6_dw1	Indicates the Flex Parser ID used to parse DW 1 of ICMPv6 header. Valid only when flex_icmpv6_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
	11:8	flex_parser_id_icmpv6_dw0	Indicates the Flex Parser ID used to parse DW 0 of ICMPv6 header. Valid only when flex_icmpv6_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
	7:4	flex_parser_id_outer_first_mpls_over_gre	Indicates the Flex Parser ID used to parse MPLS over GRE header. Valid only when flex_cw_mpls_over_gre_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
	3:0	flex_parser_id_outer_first_mpls_over_udp_label	Indicates the Flex Parser ID used to parse MPLS over UDP header. Valid only when flex_cw_mpls_over_udp_enabled in HCA_CAP.flex_parser_protocols is set.		No [FW]
DCh	31:16	max_num - match_definer	Maximum number of Match Definer objects that can be supported. See Section 8.27.28, "Match Definer Object," on page 611		No [FW]
	15:0	sf_base_id	Base sub-function identifier. This is also the base vport number of the Sub-Function on eSwitch. Valid only when HCA_CAP.sf==1.		No [FW]

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
E0h	23:0	num_total_dynamic_vf_msix	Number of MSI-X vectors that can be distributed to VFs dynamically by the VF resource manager. Value 0x0 indicates that MSI-X vector assignment is done by the device. 0 means invalid. [Internal] By default, CURRENT value is 0x0, max value is non-zero in case the feature is supported. The driver needs to set a non-zero value to current in order to activate.	RW	
E4h	11:0	dynamic_msix_table_size	Number of MSI-X vectors allocated. Value must be in range defined by min_dynamic_vf_msix_table_size and max_dynamic_vf_msix_table_size. Valid only for SET and QUERY from another HCA, which is the vport group manager. Driver should query its own MSI-X table through PCI configuration header.		
E8h	19:16	min_dynamic_vf_msix_table_size	Minimal number of Dynamic MSI-X vectors that can be set in dynamic_msix_table_size for other HCA capabilities		
	11:0	max_dynamic_vf_msix_table_size	Maximal number of Dynamic MSI-X vectors that can be set in dynamic_msix_table_size for other HCA capabilities		

Table 1572 - HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
F0h-F4h	64	vhca_tunnel_commands	Bits mask indicates when the commands are supported to be performed as tunneled commands. Bit 0: INIT_HCA Bit 1: TEARDOWN_HCA Bit 2: ENABLE_HCA Bit 3: DISABLE_HCA Bit 4: QUERY_SPECIAL_CONTEXT Bit 5: FLOW_TABLE_COMMANDS Bit 6: QP_COMMANDS Bit 9: SET_ROCE_ADDRESS Bit 10: QUERY_ROCE_ADDRESS Bit 11: GEN_OBJ_CMD_NVME_SQ - General object command of NVME_SQ Bit 12: GEN_OBJ_CMD_NVME_CQ - General object command of NVME_CQ Bit 13: GEN_OBJ_CMD_NVME_DEVICE_EUMULATION: General object command of DEVICE_EUMULATION Bit 14: NVME_NAMESPACE Bit 15: PD		
F8h-FCh	64	match_definer_format_supported	Bitmask indicates the supported Match Definer Formats. See Section 8.27.28, "Match Definer Object," on page 611		No [FW]

29.3.4.2 ODP Capabilities

Querying for ODP capabilities provides detailed ODP capabilities. ODP support for each operation is indicated on a per-transport basis and applies to direct MKeys only. Indirect MKeys and memory windows are currently not supported, and ODP behavior associated with them is unde-

fined. In addition, ODP does not currently support signature operations. Using ODP alongside signature results in undefined behavior.

Table 1573 - ODP Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-04h
sig																																08h
																																0Ch
rc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)																																10h
uc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)																																14h
ud_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)																																18h
xrc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)																																1Ch
dc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)																																20h
																																24h-FCh

Table 1574 - ODP Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	sig	ODP supported with signature hand-over operations. See Section 17.4, “Signature Handover Programming,” on page 898.	
10h	31:0	rc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)	RC transport ODP capabilities.	

Table 1574 - ODP Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
14h	31:0	uc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)	UC transport ODP capabilities.	
18h	31:0	ud_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)	UD transport ODP capabilities.	
1Ch	31:0	xrc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)	XRC transport ODP capabilities.	
20h	31:0	dc_odp_caps (See Table 1575, “ODP per Transport Service Capabilities Layout,” on page 1727)	DC transport ODP capabilities.	

Table 1575 - ODP per Transport Service Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
send	receive	write	read	atomic	rmp																										0h	

Table 1576 - ODP per Transport Service Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
0h	31	send	Send operations are supported. [Internal] Device must report 0 in current capabilities for all the transport services except of RC where they must be 1.	RW
	30	receive	Receive operations are supported. [Internal] Device must report 0 in current capabilities for all the transport services except of RC where they must be 1.	RW
	29	write	RDMA write operations are supported. SW must set same configuration for write and read fields. [Internal] Device must report 0 in current capabilities for all the transport services except of RC where they must be 1.	RW
	28	read	RDMA read operations are supported. SW must set same configuration for write and read fields. [Internal] Device must report 0 in current capabilities for all the transport services except of RC where they must be 1.	RW
	27	atomic	Atomic operations are supported. SW must set same configuration for write and read fields. [Internal] Device must report 0 in current capabilities for all the transport services except of RC where they must be 1.	RW
	26	rmp	Receive on RMP is supported. [Internal] Device must report 0 in current capabilities of rmp field for all the transport services.	RW

29.3.4.3 Atomic Capabilities

Querying for Atomic capabilities provides detailed atomic capabilities. Before querying these capabilities, software must make sure that the atomic field in HCA_CAP is equal to 1. See Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1577 - Atomic Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h-04h

Table 1577 - Atomic Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
atomic_req_8b_endianness_mode						supported atomic_req_8B_endianness_mode_1																									08h	
																														0Ch		
																atomic_operations												10h				
																atomic_size_qp												14h				
																atomic_size_dc												18h				
																												1Ch				
																load_remote_micro_app_4byte												20h				
																load_remote_micro_app_8byte												24h				
																store_remote_micro_app_4byte												28h				
																store_remote_micro_app_8byte												2Ch				
																												30h-4Ch				
fetch_add_pci_atomic																swap_pci_atomic												50h				
compare_swap_pci_atomic																												54h				
																												58h-FCCh				

Table 1578 - Atomic Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
08h	31:3 0	atom- ic_req_8b_endian- ness_mode	Active 8 Byte Atomic operations (standard and extended) endianness format in requester scatter. Note: 1-4 and 16-256 Byte Atomic operations and remote micro operations are always in BE. 0x0: big - requester endianness format is always Big endian for standard 8B and extended 8B atomics. 0x1: according_to_host_endianness - requester endianness format is according to HOST_ENDIANNESS register for standard 8B and extended 8B atomics. Supported only when atom-ic_req_8B_endianness_mode_1 ==1. Note that in responder, the scatter endianness is always according to HOST_ENDIANNESS register.	RW
	25	supported_ - atom- ic_req_8B_endian ness_mode_1	Atomic requester 8B atomics mode 0x1 support.	
10h	15:0	atomic_operations	Bit mask for atomic operations supported, bit equal to 1 means operations is supported. bit 0: Atomic_Compare_and_Swap_supported bit 1: Atomic_Fetch_and_Add_supported bit 2: Atomic_Masked_Compare_and_Swap_supported - extended atomic operation bit 3: Atomic_Masked_Fetch_and_Add_supported - extended atomic operation bit 4: remote_micro_applications_supported	
14h	15:0	atomic_size_qp	Atomic sizes supported for all qps transport services except of DC. bit 0: byte_1 bit 1: bytes_2 bit 2: bytes_4 bit 3: bytes_8 bit 4: bytes_16 bit 5: bytes_32 bit 6: bytes_64 bit 7: bytes_128 bit 8: bytes_256	

Table 1578 - Atomic Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
18h	15:0	atomic_size_dc	Atomic sizes supported for DC qps. bit 0: byte_1 bit 1: bytes_2 bit 2: bytes_4 bit 3: bytes_8 bit 4: bytes_16 bit 5: bytes_32 bit 6: bytes_64 bit 7: bytes_128 bit 8: bytes_256	
20h	31:0	load_remote_micro_app_4byte	Bitmask indicates which load remote micro operations of 4 byte operands size are supported. Bits x for operation code ==x. See Section 25, "Remote Micro-Applications," on page 1042	
24h	31:0	load_remote_micro_app_8byte	Bitmask indicates which load remote micro operations of 8 byte operands size are supported. Bits x for operation code ==x. See Section 25, "Remote Micro-Applications," on page 1042	
28h	31:0	store_remote_micro_app_4byte	Bitmask indicates which store remote micro operations of 4 byte operands size are supported. Bits x for operation code ==x. See Section 25, "Remote Micro-Applications," on page 1042	
2Ch	31:0	store_remote_micro_app_8byte	Bitmask indicates which store remote micro operations of 8 byte operands size are supported. Bits x for operation code ==x. See Section 25, "Remote Micro-Applications," on page 1042	
50h	31:16	fetch_add_pci_atomic	Bitmask indicates for which sizes of FETCH_ADD PCI Atomic are supported. Bit 0: Bytes_4 Bit 1: Bytes_8 Bit 2: Bytes_16	
	15:0	swap_pci_atomic	Bitmask indicates for which sizes of SWAP PCI Atomic are supported. Bit 0: Bytes_4 Bit 1: Bytes_8 Bit 2: Bytes_16	
54h	31:16	compare_swap_pci_atomic	Bitmask indicates for which sizes of COMPARE_SWAP PCI Atomic are supported. Bit 0: Bytes_4 Bit 1: Bytes_8 Bit 2: Bytes_16	

29.3.4.4 RoCE Capabilities

Querying for RoCE capabilities provides detailed RoCE capabilities. Before querying these capabilities, software must make sure that the RoCE field in HCA_CAP is equal to 1. (See Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1579 - RoCE Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
																																04h-0Ch
																																10h
																																14h
																																18h
																																1Ch
																																20h-FCh

Table 1580 - RoCE Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	roce_apm	If set, Automatic Path Migration is supported.	
	30	rts2rts_primary_eth_prio	If set, RTS2RTS command supports setting eth_prio in primary path. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919.	
	29	roce_rx_allow_untagged	If set, received untagged packets are allowed depending on Roce address configurations. See Table 2532, “RoCE Address Field Descriptions,” on page 2174..	
	28	rts2rts_src_addr_index_for_vlan_valid_vlan_id	If set, RTS2RTS command supports setting src_addr_index in primary path. See Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919. Note: Updating src_addr_index is allowed only if new source address index contains updates in vlan_id, vlan_valid of RoCE Address Layout. See Table 2531, “RoCE Address Layout,” on page 2173.	
	27	sw_r_roce_src_udp_port	When 0, the device sets the UDP source port for all egress RoCE v2 packets. The device keeps the source port in the range of r_roce_min_src_udp_port and r_roce_max_src_udp_port. When 1, the device supports SW defined UDP source port for RoCE v2 packet. Enable it is by setting this field to 1. [Internal] FW must report in max 1 and in current 0.	RW
	26	fl_rc_qp_when_roce_disabled	If set, RC RoCE QP creation with force loopback==1 is allowed when RoCE is disabled in nic vport context. On such case source and destination network addresses should be reserved"	
10h	19:16	l3_type	Bitmask field indicates the supported L3 types in RoCE Bit 0: Grh Bit 1: Ipv4 Bit 2: Ipv6	
	7:0	roce_version	Bitmask field indicates the supported RoCE Protocol Versions. RoCE Versions Bitmask Fields Bit 0: version_1_0 - RoCE (v1) Bit 1: version_1_5 - R-RoCE IP Based (v1.5) Bit 2: version_2_0 - R-RoCE UDP Based (v2) When bitmask is zero RoCE is not supported. RoCE Address table is allowed to contain only supported RoCE versions and only those packets which belong to supported RoCE protocols can be delivered to RoCE QPs.	

Table 1580 - RoCE Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
14h	15:0	r_roce_dest_udp_port	For R-RoCE over UDP, UDP.dport.	
18h	31:16	r_roce_max_src_udp_port	Port UDP source port provided by SW must be in the range [r_roce_min_src_udp_port- r_roce_max_src_udp_port]	
	15:0	r_roce_min_src_udp_port	UDP source port provided by SW must be in the range [r_roce_min_src_udp_port- r_roce_max_src_udp_port],	
1Ch	15:0	roce_address_table_size	Size of RoCE address table. Software allowed to access entries [0..roce_address_table_size-1]	

29.3.4.5 Networking Offload Capabilities

This data structure is the output of QUERY_HCA_CAP to Query capabilities of Ethernet, **IPoIB** and **EoIB** networks offloads. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667. Before querying these capabilities, software must make sure that the Ethernet field in HCA_CAP is equal to 1.

Table 1581 - Per Protocol Networking Offload Capabilities Layout

Offset	00h	04h
31	tunnel Stateless vxlan	tunnel Stateless geneve rx
30	tunnel Stateless gre	max geneve opt len
29	tunnel lro vxlan	
28	tunnel lro gre	
27	tunnel lso const out ip id	
26	enhanced multi pkt send wqe	
25	scatter fcs	
24	reg_umr_sq	
23	rss_ind_tbl_cap	
22	wqe_inline_mode	
21	multi_pkt_send_wqe	
20	max_lso_cap	
19		
18		
17		
16		
15		
14		
13		
12		
11		
10		
9		
8		
7		
6		
5		
4		
3		
2		
1		
0		
31	self_lb_uc	tunnel Stateless gtp
30	self_lb_mc	insert trailer
29	self_lb_en_modifiable	tunnel Stateless ip over ip
28	wqe_vlan_insert	tunnel Stateless ipv4 over vxlan
27	lro_max_msg_sz_mode	tunnel Stateless vxlan gre
26	lro time stamp	tunnel Stateless mpls over gre
25	lro psh flag	tunnel Stateless mpls over udp
24	lro cap	tunnel Stateless geneve tx
23	vlan strip	ecg checksum full
22	csu_m_cap	swp lso
21		swp csu_m
20		swp

Table 1581 - Per Protocol Networking Offload Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																lro_min_mss_size																08h
																																0Ch-2Ch
lro_timer_supported_periods[0]																																30h
lro_timer_supported_periods[1]																																34h
lro_timer_supported_periods[2]																																38h
lro_timer_supported_periods[3]																																3Ch
																																40h-FCh

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	csum_cap	Checksum Offload capability is supported	
	30	vlan_strip	VLAN stripping offload capability is supported	
	29	lro_cap	LRO hardware offload is supported	
	28	lro_psh_flag	When set, the adapter supports LRO for segments with a TCP PSH bit enabled. Reserved when lro_cap = 0 .	
	27	lro_time_stamp	When set, the adapter supports LRO for segments with a TCP timestamp option. Reserved when lro_cap = 0.	
	26:25	lro_max_msg_sz_mode	lro_max_message_size_mode reports which LRO max message size mode the device supports. 0x0: start_from_TCP_header - TIR. lro_max_message_size field sets max LRO IP payload size (TCP header + TCP payload). [internal - Shomron mode] 0x1: start_from_L2_header - TIR.lro_max_message_size field sets max LRO message size starting from L2 headers (L2 + L3 + TCP headers + TCP payload). Reserved when lro_cap = 0 [internal - Dotan mode] The device allows to limit per TIR the maximum message size LRO is allowed to aggregate. This limit can be set per TIR via lro_max_message_size field. See Table 116, “TIR Context Format,” on page 374.	
	24	wqe_vlan_insert	If set, WQE Vlan insertion is supported. See Section 4.3.10, “VLAN Insertion/Stripping,” on page 201.	
	23	self_lb_en_modifiable	If set, self_lb_en in TIR Context is modifiable. See Table 120, “MODIFY_TIR Bitmask,” on page 380.	
	22	self_lb_mc	If set, self-loopback for multicast is supported. When self-multicast loopback is supported it can be enabled per TIR via TIR.self_lb_en field. For more details, See Table 116, “TIR Context Format,” on page 374.	
	21	self_lb_uc	If set, self-loopback for unicast is supported. When self-unicast loopback is supported, it can be enabled per TIR via TIR.self_lb_en field. For more details, See Table 116, “TIR Context Format,” on page 374.	

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	20:16	max_lso_cap	Log (base2) of the maximum LSO message (TCP-payload) supported. 0 - LSO is not supported.	
	15:14	multi_pkt_send_wqe	Multi Packet Send Wqe 0x0: not_supported. 0x1: Privileged - Supported for privileged SQs only. See Section 4.3.4, "Multi-Packet Send WQE Offload," on page 191.	
	13:12	wqe_inline_mode	Wqe inline mode 0: L2 - min inline mode is L2 1: vport_context - min inline mode is according to nic_vport_context for ETH port and hca_vport_context for IB port. 2: not_required - inline not required	
	11:8	rss_ind_tbl_cap	Log (base2) of the maximum RSS indirection table size is supported. 0 - RSS is not supported.	
	7	reg_umr_sq	If set, Reg UMR SQ is supported. See Table 138, "SQ Context Format," on page 404	
	6	scatter_fcs	If set, setting RQ.scatter_fcs is supported. Valid only for Ethernet. See Table 124, "RQ Context Format," on page 385.	

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	5	enhanced_multi_pkt_send_wqe	If set, Enhanced multi packet send WQE is supported. See Section 4.3.5, “Enhanced Multi-Packet Send WQE Offload,” on page 193	
	4	tunnel_lso_const_out_ip_id	If set IP Identification field of the outer IP header of segmented by LSO tunneling packet is not incremented.	RW
	3	tunnel_lro_gre	If set, LRO for GRE tunneled packets is supported.	
	2	tunnel_lro_vxlan	If set, LRO for VXLAN L2 or L3 tunneled packets is supported.	
	1	tunnel_stateless_gre	If set, Stateless Offloads for GRE L2 or L3 tunneled packets are supported. The following stateless offloads are supported for the GRE tunneled (inner) packets when this flag is on: RSS, RX & TX Checksum Offloads, LSO, Flow Steering, VLAN insertion.	
	0	tunnel_stateless_vxlan	If set, Stateless Offloads for VXLAN tunneled (inner) packets are supported. The following stateless offloads are supported for the VXLAN tunneled (inner) packets when this flag is on: RSS, RX & TX Checksum Offloads, LSO, Flow Steering, VLAN insertion.	
04h	31	swp	If set, Software Parsing is supported. See Section 4.3.3, “Software Parsing - SWP,” on page 191.	
	30	swp_csum	If set, Checksum offload with Software Parsing is supported. See Section 4.3.3, “Software Parsing - SWP,” on page 191.	
	29	swp_lso	If set, LSO offload with Software Parsing is supported. See Section 4.3.3, “Software Parsing - SWP,” on page 191.	

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
04h	28	cqe_checksum_full	When set, the device checksum calculation reported in the CQE is modified to include all packet fields above layer 2. See "4.3.1 Checksum Offload"	
	27	tunnel_stateless_geneve_tx	If set, Transmit Stateless Offloads for Geneve tunneled packets are supported. The following stateless are supported for the inner packets: TX Checksum Offloads, LSO, Flow Steering.	

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
04h	26	tunnel_stateless_mpls_over_udp	If set, Stateless Offloads for MPLS (w/ CW) over UDP tunneled packets are supported. The following offloads are supported for the inner packets: RSS, RX & TX Checksum Offloads, LSO, Flow Steering. In addition, if packet Decapsulation is supported, it is also supported for this kind of encapsulation.	
	25	tunnel_stateless_mpls_over_gre	If set, Stateless Offloads for MPLS (w/ CW) over GRE tunneled packets are supported. The following offloads are supported for the inner packets: RSS, RX & TX Checksum Offloads, LSO, Flow Steering. In addition, if packet Decapsulation is supported, it is also supported for this kind of encapsulation.	
	24	tunnel_stateless_vxlan_gpe	If set, Stateless Offloads for VXLAN-GPE tunneled packets are supported. The following offloads are supported for the inner packets: RSS, RX & TX Checksum Offloads, LSO, Flow Steering. In addition, if packet Decapsulation is supported, it is also supported for this kind of encapsulation.	
	23	tunnel_stateless_ipv4_over_vxlan	If set, Stateless Offloads for IPv4 over VXLAN tunneled packets are supported. The following offloads are supported for the inner packets: RSS, RX & TX Checksum Offloads, LSO, Flow Steering. In addition, if packet Decapsulation is supported, it is also supported for this kind of encapsulation.	
	22	tunnel_stateless_ip_over_ip	If set, Stateless Offloads for IP over IP tunneled packets are supported. The following offloads are supported for the inner packets: RSS, RX & TX Checksum Offloads, LSO, Flow Steering. In addition, if packet Decapsulation is supported, it is also supported for this kind of encapsulation.	
	21	insert_trailer	When set Inserting a packet trailer through WQE is supported (in ETH Segment). See Table 47, “Eth Segment Format,” on page 318	
	20	tunnel_stateless_gtp	Of set, Stateless Offloads for GTP tunneled packets are supported.	
	15:8	max_vxlan_udp_ports	Maximum number of supported vxlan udp ports that can be added by ADD_VXLAN_UDP_DPORT command. Note that 0 indicates 4 ports.	

Table 1582 - Per Protocol Networking Offload Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
04h	1	max_geneve_opt_len	Maximum geneve_opt_len supported. Valid only when tunnel_stateless_geneve_rx==1. 0x0: 14DW 0x1: 63DW SW must filter GENEVE packets with geneve_opt_len greater than the supported values using Flow Table mechanism. [Internal] If max_geneve_opt_len==0, the device parses the geneve inner headers wrongly in case the value of the OptLen field of the geneve header in the packet is greater than 14. This wrong inner headers parsing will yield un-expected behavior in inner checksum reporting in the CQE, in flow steering rules that match inner headers and for TIRs that use the rx_hash_field_selector_inner fields. To protect from this un-expected behavior the driver may use the geneve_opt_len field of “Table 1233 – Flow Table Entry Match Set Misc Parameters Format” <add cross reference> (with appropriate masks) to filter out (drop) geneve packets with OptLen>14. For debug-ability, the driver may attach a flow counter to these filter-out rules in order to count these packets that were dropped.	
	0	tunnel_stateless_geneve_rx	If set, the following receive Stateless Offloads for Geneve tunneled (inner) packets are supported: RSS, Checksum Offloads, Flow Steering. On transmit side, Checksum and LSO for Geneve tunneled (inner) packets may be achieved using the Software Parsing mechanism. See Section 4.3.3, “Software Parsing - SWP,” on page 191	
08h	15:0	lro_min_mss_size	Minimal TCP payload size required for LRO. Must be >= 1. Reserved when lro_cap = 0.	
30h-3Ch	31:0	lro_timer_supported_periods[4]	Array of supported LRO timer periods in microseconds. The supported timers are organized in ascending order. When requested timer's period is N timer's expiration period can fluctuate between N and 2N. Reserved when lro_cap = 0.	

29.3.4.6 Flow Table Capabilities

This data structure is the output of QUERY_HCA_CAP to query capabilities of Flow Table. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667.

NIC Flow Tables are supported when HCA_CAP.nic_flow_table==1.

NIC Flow Tables capabilities are described in Table 1583, “Flow Table NIC Capabilities Layout”.

Table 1583 - Flow Table NIC Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nic_rx_multi_path_tirs	nic_rx_multi_path_tirs	allow_sniffer_and_nic_rx_shared_tir	nic_rx_flow_tag_count_en	nic_rx_flow_tag_multipath_en	ttl_checksum_correction	nic_rx_rdma_fwd_tir	sw_owner_reformat_supported																									00h
encap_general_header											log_max_packet_reformat_context																					04h
																																08h-3Ch
flow_table_properties_nic_receive (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																40h-7Ch
flow_table_properties_nic_receive_rdma (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																80h-BCh
flow_table_properties_nic_receive_sniffer (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																C0h-FCh
flow_table_properties_nic_transmit (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																100h-13Ch
flow_table_properties_nic_transmit_rdma (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																140h-17Ch

Table 1583 - Flow Table NIC Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
flow_table_properties_nic_transmit_sniffer (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																180h-1BCCh
																																1C0h-1FCCh
header_modify_nic_receive (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)																																200h-27Ch
																																280h-37Ch
header_modify_nic_transmit (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)																																380h-3FCh
sw_steering_nic_rx_action_drop_icm_address																																400h-404h
sw_steering_nic_tx_action_drop_icm_address																																408h-40Ch
sw_steering_nic_tx_action_allow_icm_address																																410h-414h
																																418h-FFCh

Table 1584 - Flow Table NIC Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
00h	31	nic_rx_multi_path_tirs	If set, this NIC Receive Flow Table supports multiple processing paths for TIRs. See Section 8.18.2.4, “Multi-Processing Paths,” on page 417		
	30	nic_rx_multi_path_tirs_fts	If set, this NIC Receive Flow Table supports multiple processing paths for TIRs and Flow tables. See Section 8.18.2.4, “Multi-Processing Paths,” on page 417 Note: Regardless of whether the destination list includes TIRs or not, the last Flow Table in the destination list is not required to have a level equal or greater than 64 like other Flow Tables in the list.		
	29	allow_sniffer_and_nic_rx_shared_tir	If set, the same TIR can be used by multiple Flow Tables of the following different types: nic receive, receive sniffer, transmit sniffer.		
	28	nic_rx_flow_tag_count_en	When set, NIC RX table supports both flow_tag and <i>Count</i> action for the same flow.		
	27	nic_rx_flow_tag_multi_path_en	When set, NIC RX table supports setting flow_tag for Flows beyond Multi-Processing split. See Section 8.18.2.9, “Flow Tagging,” on page 418		
	26	ttr_checksum_correction	If set, CQE checksum field may be used even if TTL field was modified by Flow steering tables.		
	25	nic_rx_rdma_fwd_tir	When set, TIR can be used as a destination in NIC Receive RDMA table		
	24	sw_owner_reformat_supported	If set, SW owner flow table can support reformat action. [Internal] this is a WA capability where old FW didn't support reformat with sw steering.		No [FW]
	7:0	nic_receive_max_steering_depth	Indicates a limit to the longest path of packet traversing through the NIC receive steering table. The field is given in device specific units. Value 0x0 indicates unlimited depth.		Yes [Trust]

Table 1584 - Flow Table NIC Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
04h	31	encap_general_header	If set, encapsulation of Generalheader is supported. Supporting general header encapsulation isn't necessary mean offload and parsing is also supported. So SW need to make sure the encapsulated protocol can be parsed add offloaded is needed.		Yes [Trust]
	20:16	log_max_packet_reformat_context	Log (base 2) of the maximum packet reformat contexts that can be created by ALLOC_PACKET_REFORMAT_CONTEXT..		
	9:0	max_encap_header_size	The maximum size of the encapsulation header supported by ALLOC_PACKET_REFORMAT_CONTEXT		
40h-7Ch	512	flow_table_properties_nic_receive (See Table 1587, "Flow Table Properties Layout," on page 1753)	Capabilities and properties of NIC Receive Flow Tables		
80h-BCh	512	flow_table_properties_nic_receive_rdma (See Table 1587, "Flow Table Properties Layout," on page 1753)	Capabilities and properties of NIC Receive RDMA Flow Tables		
C0h-FCh	512	flow_table_properties_nic_receive_sniffer (See Table 1587, "Flow Table Properties Layout," on page 1753)	Capabilities and properties of NIC Receive Sniffer Flow Tables		
100h-13Ch	512	flow_table_properties_nic_transmit (See Table 1587, "Flow Table Properties Layout," on page 1753)	Capabilities and properties of NIC Transmit Flow Tables		
140h-17Ch	512	flow_table_properties_nic_transmit_rdma (See Table 1587, "Flow Table Properties Layout," on page 1753)	Capabilities and properties of NIC Transmit RDMA Flow Tables		

Table 1584 - Flow Table NIC Capabilities Field Descriptions

Offset	Bits	Name	Description	Access of current capabilities	Modifiable by vhca_resource_manager
180h-1BCh	512	flow_table_properties_nic_transmit_sniffer (See Table 1587, “Flow Table Properties Layout,” on page 1753)	Capabilities and properties of NIC Transmit Sniffer Flow Tables		
200h-27Ch	1024	header_modify_nic_receive (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)	Header modify capabilities of NIC Receive Flow Tables		
380h-3FCh	1024	header_modify_nic_transmit (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)	Header modify capabilities of NIC Transmit Flow Tables		
400h-404h	64	sw_steering_nic_rx_action_drop_icm_address	ICM address representing a drop action. Should be used by SW managed steering for efficient packet drops in NIC Rx Flow Table Type.		
408h-40Ch	64	sw_steering_nic_tx_action_drop_icm_address	ICM address representing a drop action. Should be used by SW managed steering for efficient packet drops in NIC Tx Flow Table Type.		
410h-414h	64	sw_steering_nic_tx_action_allow_icm_address	ICM address representing a Allow action. Should be used by SW managed steering NIC Tx Flow Table Type.		

E-switch Flow Tables capabilities is described in Table 1585, “Flow Table E-switch Capabilities Layout”.
E-switch Flow Tables is supported when HCA_CAP.eswitch_manager==1.

Table 1585 - Flow Table E-switch Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fdb_to_vport_reg_c_id																				ingress_acl_flow_counter_for_drop_actions	esw_fdb_modify_header_fwd_to_table	esw_fdb_ipv4_ttl_modify	flow_source		extended_fdb_flow_counter_en	multi_fdb_encap	egress_acl_forward_to_vport	fdb_multi_path_to_table	ingress_acl_forward_to_vport	fdb_flow_counter_en		00h
																																04h-3Ch
flow_table_properties_nic_esw_fdb (See Table 1587, “Flow Table Properties Layout,” on page 1753) <u>TBD - do we want a different capability opmod to separate NIC from switch ?</u>																																40h-7Ch
flow_table_properties_esw_acl_ingress (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																80h-BCh
flow_table_properties_esw_acl_egress (See Table 1587, “Flow Table Properties Layout,” on page 1753)																																C0h-FCh
header_modify_esw_fdb (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)																																100h-17Ch
header_modify_esw_acl_ingress (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)																																180h-1FCh
header_modify_esw_acl_egress (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)																																200h-27Ch

Table 1585 - Flow Table E-switch Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															280h-2FCh	
sw_steering_fdb_action_drop_icm_address_rx																															300h-304h	
sw_steering_fdb_action_drop_icm_address_tx																															308h-30Ch	
sw_steering_uplink_icm_address_rx																															310h-314h	
sw_steering_uplink_icm_address_tx																															318h-31Ch	
																															320h-FFCh	

Table 1586 - Flow Table E-switch Capabilities Fields

Offset	Bits	Name	Description	Access
00h	31:24	fdb_to_vport_reg_c_id	Bit map per register_c ID indicating whether this register can be used to pass metadata to Vport Rx Flow Table. The ESW VPORT Context indicates whether this feature is supported and which Registers to pass. Set bit means this register is supported. Bit 0: REG_C_0 Bit 1: REG_C_1 Bit 2: REG_C_2 Bit 3: REG_C_3 Bit 4: REG_C_4 Bit 5: REG_C_5 Bit 6: REG_C_6 Bit 7: REG_C_7	
	11	ingress_acl_flow_counter_for_drop_actions	If set, Ingress ACL Flow Table type may include a Flow Counter for flow with Drop actions.	
	10	esw_fdb_modify_header_fwd_to_table	When set, FDB table supports Modify Header action and Forward to Flow Table action in the same Flow.	
	9	esw_fdb_ipv4_ttl_modify	When set, the device supports modifying the TTL even for Flows forwarding packets from uplink to non-uplink vports.	
	8	flow_source	When set, the flow_source field of in the Flow Context can be used to indicate the known origin of packets matching this Flow.	
	6	extended_fdb_flow_counter_en	Describes the limitations under which Flow Counters are supported in the FDB Flow Table besides extended_fdb_flow_counter_en. See Section 8.18.4.7, “E-Switch Forwarding Data Base (FDB),” on page 430.	
	5	multi_fdb_encap	If set, a Flow Table with enabled Reformat/decapsulate actions may be created when not all VFs are down, if another table with these actions enabled already exists.	
	4	egress_acl_forward_to_vport	When set, the Egress ACL table supports the Forward action to another VPort.	
	3	fdb_multi_path_to_table	If set, the FDB table supports the Forward action with a destination list that includes a Flow Table	
	2	ingress_acl_forward_to_vport	If set, the Ingress ACL table supports the Forward action to another VPort	
	1:0	fdb_flow_counter_en	Describes the limitations under which Flow Counters are supported in the FDB Flow Table beside extended_fdb_flow_counter_en. See Section 8.18.4.7, “E-Switch Forwarding Data Base (FDB),” on page 430.	

Table 1586 - Flow Table E-switch Capabilities Fields

Offset	Bits	Name	Description	Access
40h-7Ch	512	flow_table_properties_nic_esw_fdb(See Table 1587, “Flow Table Properties Layout,” on page 1753)	Capabilities and properties of E-Switch FDB Flow Tables	
80h-BCh	512	flow_table_properties_esw_acl_ingress (See Table 1587, “Flow Table Properties Layout,” on page 1753)	Capabilities and properties of E-Switch Ingress ACL Flow Tables	
C0h-FCh	512	flow_table_properties_esw_acl_egress (See Table 1587, “Flow Table Properties Layout,” on page 1753)	Capabilities and properties of E-Switch Egress ACL Flow Tables	
100h-17Ch	1024	header_modify_esw_fdb (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)	Header modify capabilities of E-Switch FDB Flow Tables	
180h-1FCh	1024	header_modify_esw_acl_ingress (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)	Header modify capabilities of E-Switch Ingress ACL Flow	
200h-27Ch	1024	header_modify_esw_acl_egress (See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760)	Header modify capabilities of E-Switch Egress ACL Flow	
300h-304h	64	sw_steering_fdb_action_drop_icm_address_rx	ICM address in FDB Rx steering domain performing a Drop action. valid only if FDB table supports SW owner, and the value is non-zero	
308h-30Ch	64	sw_steering_fdb_action_drop_icm_address_tx	ICM address in FDB Tx steering domain performing a Drop action. valid only if FDB table supports SW owner, and the value is non-zero	

Table 1586 - Flow Table E-switch Capabilities Fields

Offset	Bits	Name	Description	Access
310h-314h	64	sw_steering_uplink_icm_address_rx	ICM address in FDB Rx steering domain sending the packet to the Uplink. valid only if FDB table supports SW owner, and the value is non-zero	
318h-31Ch	64	sw_steering_uplink_icm_address_tx	ICM address in FDB Tx steering domain sending the packet to the Uplink. valid only if FDB table supports SW owner, and the value is non-zero	

Table 1587 - Flow Table Properties Layout

Offset	00h	04h	08h	0Ch
0	wqe_based_flow_update	max_ft_level	metadata_reg_a_width	log_max_ft_num
1	sw_owner_v2			
2	ipsec decrypt			
3	ipsec encrypt			
4	forward_vhca_tx			
5	forward_vhca_rx			
6	reformat and fwd to table			
7	termination table			
8	table miss action domain			
9	ip type			
10	ignore flow level	max_modify_header_actions	metadata_reg_b_width	
11	reformat_and_modify_action			
12	reformat_l2_to_l3_tunnel			
13	reformat_l3_tunnel_to_l2			
14	sw_owner			
15	modify_and_vlan_action			
16	reformat_and_vlan_action	log_max_modify_header_context	log_max_flow_sampler_num	
17	push_vlan_2			
18	pop_vlan_2			
19	fpga_vendor_acceleration			
20	push_vlan			
21	pop_vlan			
22	reset root to default	log_max_modify_header_context	log_max_flow_sampler_num	
23	decap			
24	reformat			
25	flow table modify			
26	identified miss table			
27	modify root			
28	flow modify en	log_max_ft_size		
29	flow counter			
30	flow_tag			
31	ft support			
	termination table raw traffic			

Table 1587 - Flow Table Properties Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																log_max_flow_counter								log_max_destination								10h
																								log_max_flow								14h
wqe_based_flow_update_action																																18h
																																1Ch
ft_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)																																20h-2Ch
ft_field_bitmask_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)																																30h-3Ch

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	ft_support	When set, this Flow Table type is supported.	
	30	flow_tag	If set, indicates that flow_tag which is reported in CQE is supported for this type flow table.	
	29	flow_counter	When set, this Flow Table type supports associating flow counters to its flows.	
	28	flow_modify_en	When set, this Flow Table type supports modifying flow entries. See Section 8.18.3.10, “Redefining a Flow,” on page 426.	
	27	modify_root	When set, this Flow Table type supports dynamic modification of the root Flow Table.	
	26	identified_miss_table	If set, forward to identified miss table when creating new flow table is supported. See Table 29.18.1, “CREATE_FLOW_TABLE - Allocate a New Flow Table,” on page 1986.	
	25	flow_table_modify	If set, MODIFY_FLOW_TABLE is supported. See Section 29.18.2, “MODIFY_FLOW_TABLE - Modify a Flow Table,” on page 1991.	
	24	reformat	If set, Reformat action is supported.	
	23	decap	If set, decapsulation action is supported.	
	22	reset_root_to_default	If set, indicates if reset root table to default behavior is supported. See Table 2113, “SET_FLOW_TABLE_ROOT - Input Structure Field Descriptions,” on page 1996	

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
00h	21	pop_vlan	When set, this Flow Table supports <i>Pop VLAN</i> action (POP_VLAN)	
	20	push_vlan	When set, this Flow Table supports <i>Push VLAN</i> action (PUSH_VLAN)	
	19	fpga_vendor_acceleration	If set, FPGA vendor acceleration action is supported.	
	18	pop_vlan_2	When set, this Flow Table supports <i>Pop VLAN_2</i> action (POP_VLAN_2) pop_vlan_2 is illegal if pop_vlan is not set. TBD: Ahmad	
	17	push_vlan_2	When set, this Flow Table supports <i>Push VLAN_2</i> action (PUSH_VLAN_2) push_vlan_2 always happens <u>after</u> push_vlan, so push_vlan_2_tag will be the <u>outermost</u> . push_vlan_2 is illegal if not setting also push_vlan TBD: Ahmad	

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
00h	16	reformat_and_vlan_action	When set, this Flow Table type supports setting Reformat related and VLAN related actions for the same FTE (if both actions are supported).	
	15	modify_and_vlan_action	When set, this Flow Table type supports setting Modify Headers and VLAN related actions for the same FTE (if both actions are supported).	
	14	sw_owner	When set, this flow table type supports using SW managed Flow Tables. Note that if sw_owner_v2 is set, this field should be ignored	
	13	reformat_l3_tunnel_to_l2	If set, reformatting the packet from tunneled L3 packet to an L2 packet is supported (remove the tunnel and add L2 header)	
	12	reformat_l2_to_l3_tunnel	If set, reformatting the packet from L2 packet to a tunneled L3 packet is supported (remove the L2 header and add tunnel header)	
	11	reformat_and_modify_action	When set, this Flow Table type supports setting <i>Modify Headers</i> and <i>Reformat</i> related actions for the same FTE (if both actions are supported).	
	10	ignore_flow_level	If set, A Flow Table Entry may point to a Flow Table with equal or lower Flow Level.	
	9	ip_type	If set, ip_type in CRAETE_FLOW_GROUP is supported.	
	8	table_miss_action_domain	When set, the Flow Table type supports defining a miss behavior that will direct the packets to a different Table Type domain	
	7	termination_table	When set, this table type supports creating Flow Tables marked as Termination.	
	6	reformat_and_fwd_to_table	When set, the flow table supports Reformat or Decapsulate actions in conjunction with Forward to Flow Table action for the same rule. Valid only if the table also supports Reformat or Decapsulate operations. In any case, programmer should not attempt to Reformat/Decapsulate the same packet more than once.	
	5	forward_vhca_rx	If set, the flow table supports VHCA_RX destination type. i.e. forwarding to other vhca receive root.	
	4	forward_vhca_tx	If set, the flow table supports VHCA_TX destination type. i.e. forwarding to other vhca transmit root.	

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
00h	3	ipsec_encrypt	When set, this table type supports IPSEC Encryption operation by IPSEC_OFFLOAD action	
	2	ipsec_decrypt	When set, this table type supports IPSEC Decryption operation by IPSEC_OFFLOAD action	
00h	1	sw_owner_v2	When set, this flow table type supports using SW managed Flow Tables. The Flow Table format version is indicated by QUERY_HCA_CAP.steering_format_version. In addition if this field is set sw_owner field should be ignored	
	0	wqe_based_flow_update	When set, this table type supports updating Flows in Flow Tables using a WQE.	
04h	31	termination_table_raw_traffic	When set, this table type supports creating Flow Tables marked as Termination also for raw traffic. i.e. for traffic that will not have modify headers	
	30	vlan_and_fwd_to_table	<u>When set, the flow table supports Pop VLAN or Push VLAN actions in conjunction with Forward action to a Flow Table in the same Entry. In any case, programmer should not attempt to push or pop more than one VLAN header for the same packet. Valid only if the table also supports Push VLAN or Pop VLAN actions.</u>	
	29:24	log_max_ft_size	Log (base2) of the Flow Table size	
	23:16	log_max_modify_header_context	Log (base 2) of the number of Modify Header Contexts supported for this type	
	15:8	max_modify_header_actions	The maximal number of actions that can be allocated by ALLOC_MODIFY_HEADER_CONTEXT. 0 means modify header not supported.	
	7:0	max_ft_level	Maximal value for the Flow Table level.	

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	reformat_add_esp_transport	when set, the relevant packet reformat operation is supported	
	30	reformat_l2_to_l3_esp_tunnel	when set, the relevant packet reformat operation is supported	
	29	reformat_add_esp_transport_over_udp	when set, the relevant packet reformat operation is supported	
	28	reformat_del_esp_transport	when set, the relevant packet reformat operation is supported	
	27	reformat_l3_esp_tunnel_to_l2	when set, the relevant packet reformat operation is supported	
	26	reformat_del_esp_transport_over_udp	when set, the relevant packet reformat operation is supported	
	23:16	log_max_flow_sampler_num	Log (base 2) of the number of Flow Sampler Objects supported for this type	
	15:8	metadata_reg_b_width	Number of supported bits in metadata_reg_b, starting at bit 0. If metadata_reg_b is not supported, this field is reserved (as indicated in Flow Table Field Supported).	
	7:0	metadata_reg_a_width	Number of supported bits in metadata_reg_a, starting at bit 0. If metadata_reg_a is not supported, this field is reserved (as indicated in Flow Table Field Supported).	
0Ch	7:0	log_max_ft_num	Log (base 2) of the number of Flow Tables supported for this type	
10h	15:8	log_max_flow_counter	Log (base 2) of the maximal number of flow counters in a single flow with <i>Count</i> action.	
	7:0	log_max_destination	Log (base 2) of the maximal number of destinations in a single Flow with <i>Forward</i> action	
14h	7:0	log_max_flow	Log (base 2) of the total number of flows supported for this type. The number of flows is the total over all Flow Tables of this type.	

Table 1588 - Flow Table Properties Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:0	wqe_based_flow_update_action	bitmap per Action ID, when set this action is supported for Flow Table Entry definition: bit 0: NOP bit 1: Transmit bit 2: inline_qpn bit 3: qp_list bit 4: Iterator bit 5: copy bit 6: set bit 7: add bit 8: remove_by_size bit 9: remove_header2header bit 10: insert_with_inline bit 11: insert_with_pointer bit 12: flow_tag bit 13: Queue_ID_selection bit 14: accelerated_modify_action_list bit 15: modify_action_list bit 16: IPSec_Encryption bit 17: IPSec_Decryption bit 18: ASO bit 19: trailer bit 20: counter_id bit 21: extended_qpn bit 22: port_selection bit 23: count_on_source_gvmi bit 24: TLS bit 25: Clear	
20h-2Ch	128	ft_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)	Bit per flow table header field support.	
30h-3Ch	128	ft_field_bitmask_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)	Bit per flow table header field bitmask support.	

Table 1589 - Header Modify Capabilities Properties Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
set_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)																																00h-0Ch
																																10h-1Ch
add_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)																																20h-2Ch
																																30h-3Ch
copy_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)																																40h-4Ch
																																50h-7Ch

Table 1590 - Header Modify Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	set_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)	Indicates which packet fields can be modified by SET action.	
20h-2Ch	128	add_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)	Indicates which packet fields can be modified by ADD action.	
40h-4Ch	128	copy_action_field_support (See Table 1591, “Flow Table Fields Supported Format,” on page 1761)	Indicates which packet fields are supported as source for COPY operations.Fields are supported as destination of the copy when SET operation is supported (indicated by set_action_field_support cap).	

Table 1591 specifies which fields are supported in the Flow Table.

Table 1591 - Flow Table Fields Supported Format

Offset	00h	04h	08h	0Ch
0	source_eswitch_port	metadata_reg_a	source_sqn	metadata_reg_c_0
1	source_vhca_port	metadata_reg_b	outer_second_svlan	metadata_reg_c_1
2	outer_geneve_opt_len		outer_first_svlan	metadata_reg_c_2
3	outer_geneve_protocol_type	inner_tcp_ack_num	inner_second_svlan	metadata_reg_c_3
4	outer_geneve_oam	outer_tcp_ack_num	inner_first_svlan	metadata_reg_c_4
5	outer_geneve_vni		btb_dst_qp	metadata_reg_c_5
6	outer_vxlan_vni		outer_ipv6_hop_limit	metadata_reg_c_6
7	outer_gre_key	inner_tcp_seq_num	inner_ipv6_hop_limit	metadata_reg_c_7
8	outer_gre_protocol	outer_tcp_seq_num	outer_esp_spi	icmp_header_data
9	outer_tcp_flags	inner_tcp_flags	inner_esp_spi	icmpv6_header_data
10	outer_tcp_dport	inner_tcp_dport	outer_end_tag	icmp_type
11	outer_tcp_sport	inner_tcp_sport	outer_first_mpls_label	icmpv6_type
12	outer_udp_dport	inner_udp_dport	outer_first_mpls_exp	icmp_code
13	outer_udp_sport	inner_udp_sport	outer_first_mpls_s_bos	icmpv6_code
14	outer_ip_dscp	inner_ip_dscp	outer_first_mpls_ttl	outer_geneve_c
15	outer_ip_een	inner_ip_een	inner_first_mpls_label	outer_geneve_ver
16	outer_ip_protocol	inner_ip_protocol	inner_first_mpls_exp	
17	outer_frag	inner_frag	inner_first_mpls_s_bos	outer_bth_pkey
18	outer_dip	inner_dip	inner_first_mpls_ttl	outer_grh_gid
19	outer_sip	inner_sip	outer_first_mpls_over_gre_label	outer_grh_tclass
20	outer_ipv6_flow_label	inner_ipv6_flow_label	outer_first_mpls_over_gre_exp	outer_grh_flow_label
21	outer_second_vid	inner_second_vid	outer_first_mpls_over_gre_s_bos	outer_lrh_lid
22	outer_second_cfi	inner_second_cfi	outer_first_mpls_over_gre_ttl	gtpu_msg_flags
23	outer_second_prio	inner_second_prio	outer_first_mpls_over_udp_label	gtpu_msg_type
24	outer_ipv4_ttl	inner_ipv4_ttl	outer_first_mpls_over_udp_exp	gtpu_teid
25	outer_first_vid	inner_first_vid	outer_first_mpls_over_udp_s_bos	inner_ipv6_extension
26	outer_first_cfi	inner_first_cfi	outer_first_mpls_over_udp_ttl	outer_ipv6_extension
27	outer_first_prio	inner_first_prio	outer_vxlan_gpe_next_protocol	ipsec_next_header
28	outer_ip_version	inner_ip_version	outer_vxlan_gpe_flags	ipsec_syndrome
29	outer_ethernet_type	inner_ethernet_type	outer_vxlan_gpe_vni	outer_gre_s_present
30	outer_smac	inner_smac	geneve_tlv_option_0_exist	outer_gre_k_present
31	outer_dmac	inner_dmac	geneve_tlv_option_0_data	outer_gre_c_present

Flow Table Field Bitmask fields descriptions are shown in Table 1592.

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
00h	31	outer_dmac	
	30	outer_smac	
	29	outer_ether_type	
	28	outer_ip_version	
	27	outer_first_prio	
	26	outer_first_cfi	
	25	outer_first_vid	
	24	outer_ipv4_ttl	
	23	outer_second_prio	
	22	outer_second_cfi	
	21	outer_second_vid	
	20	outer_ipv6_flow_label	
	19	outer_sip	
	18	outer_dip	
	17	outer_frag	
	16	outer_ip_protocol	
	15	outer_ip_ecn	
	14	outer_ip_dscp	
	13	outer_udp_sport	
	12	outer_udp_dport	
	11	outer_tcp_sport	
	10	outer_tcp_dport	
	9	outer_tcp_flags	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
00h	8	outer_gre_protocol	
	7	outer_gre_key	
	6	outer_vxlan_vni	
	5	outer_geneve_vni	
	4	outer_geneve_oam	
	3	outer_geneve_protocol_type	
	2	outer_geneve_opt_len	
	1	source_vhca_port	
	0	source_eswitch_port	
04h	31	inner_dmac	
	30	inner_smac	
	29	inner_ether_type	
	28	inner_ip_version	
	27	inner_first_prio	
	26	inner_first_cfi	
	25	inner_first_vid	
	24	inner_ipv4_ttl	
	23	inner_second_prio	
	22	inner_second_cfi	
	21	inner_second_vid	
	20	inner_ipv6_flow_label	
	19	inner_sip	
	18	inner_dip	
	17	inner_frag	
	16	inner_ip_protocol	
	15	inner_ip_ecn	
	14	inner_ip_dscp	
	13	inner_udp_sport	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
04h	12	inner_udp_dport	
	11	inner_tcp_sport	
	10	inner_tcp_dport	
	9	inner_tcp_flags	
04h	8	outer_tcp_seq_num	
	7	inner_tcp_seq_num	
	4	outer_tcp_ack_num	
	3	inner_tcp_ack_num	
04h	1	metadata_reg_b	If set, metadata_reg_b match is supported in flow table. Number of bits supported in this field is reported in Flow Table Properties, See Table 1587, “Flow Table Properties Layout,” on page 1753.
	0	metadata_reg_a	If set, metadata_reg_a match is supported in flow table. Number of bits supported in this field is reported in Flow Table Properties, See Table 1587, “Flow Table Properties Layout,” on page 1753.
08h	31	geneve_tlv_option_0_data	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
08h	30	geneve_tlv_option_0_exist	
	29	outer_vxlan_gpe_vni	
	28	outer_vxlan_gpe_flags	
	27	outer_vxlan_gpe_next_protocol	
	26	outer_first_mpls_over_udp_ttl	
	25	outer_first_mpls_over_udp_s_bos	
	24	outer_first_mpls_over_udp_exp	
	23	outer_first_mpls_over_udp_label	
	22	outer_first_mpls_over_gre_ttl	
	21	outer_first_mpls_over_gre_s_bos	
	20	outer_first_mpls_over_gre_exp	
	19	outer_first_mpls_over_gre_label	
	18	inner_first_mpls_ttl	
	17	inner_first_mpls_s_bos	
	16	inner_first_mpls_exp	
	15	inner_first_mpls_label	
	14	outer_first_mpls_ttl	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
08h	13	outer_first_mpls_s_bos	
	12	outer_first_mpls_exp	
	11	outer_first_mpls_label	
	10	outer_emd_tag	If set, Embedded Meta Data L2 tag is supported in outer header (outer_emd_tag and outer_emd_tag_data[])
	9	inner_esp_spi	
	8	outer_esp_spi	
	7	inner_ipv6_hop_limit	
	6	outer_ipv6_hop_limit	
	5	bth_dst_qp	If set, Destination QP in BTH header is supported.
	4	inner_first_svlan	Inner first VLAN supports s-vlan in addition to c-vlan
	3	inner_second_svlan	Inner second VLAN supports s-vlan in addition to c-vlan
	2	outer_first_svlan	Outer first VLAN supports s-vlan in addition to c-vlan
	1	outer_second_svlan	Outer second VLAN supports s-vlan in addition to c-vlan
	0	source_sqn	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
0Ch	31	outer_gre_c_present	
	30	outer_gre_k_present	
	29	outer_gre_s_present	
	28	ipsec_syndrome	
	27	ipsec_next_header	
	26	outer_ipv6_extension	
	25	inner_ipv6_extension	
	24	gtpu_teid	
	23	gtpu_msg_type	
	22	gtpu_msg_flags	
	21	outer_lrh_lid	Both DLID and SLID
	20	outer_grh_flow_label	
	19	outer_grh_tclass	
	18	outer_grh_gid	Both DGID and SGID
	17	outer_bth_pkey	
	15	outer_geneve_ver	
	14	outer_geneve_c	

Table 1592 - Flow Table Fields Supported Fields

Offset	Bits	Name	Description
0Ch	13	icmpv6_code	
	12	icmp_code	
	11	icmpv6_type	
	10	icmp_type	
	9	icmpv6_header_data	
	8	icmp_header_data	
	7	metadata_reg_c_7	
	6	metadata_reg_c_6	
	5	metadata_reg_c_5	
	4	metadata_reg_c_4	
	3	metadata_reg_c_3	
	2	metadata_reg_c_2	
	1	metadata_reg_c_1	
	0	metadata_reg_c_0	

29.3.4.7 E-switch Capabilities

These capabilities can be used only by vport_group_manager when e-switch capabilities are supported (HCA_CAP.eswitch_manager==1).

Table 1593 - E-switch Capabilities Layout

Offset	00h
0	nic_vport_port_guid_modify
1	nic_vport_node_guid_modify
2	merged_eswitch
3	counter_eswitch_affinity
4	ecpf_vport_exists
5	fdb_single_tir_destination
6	esw_functions_changed
7	match_uplink_other_esw
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	sw_network_metadata
22	root_ft_on_other_esw
23	esw_uplink_ingress_acl
24	esw_shared_ingress_acl
25	vport_cvlan_insert_always
26	not to use
27	vport_cvlan_insert_override
28	vport_cvlan_insert_if_not_exist
29	vport_svlan_insert
30	vport_cvlan_strip
31	vport_svlan_strip

Table 1593 - E-switch Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
encap_general_header			log_max_fd- b_encap_uplink									log_max_pack- et_reformat_- context											max_encap_header_size										04h
vxlan_encap_decap											log_max- _esw_sf					esw_sf_base_id														08h			
																																0Ch-FCh	

Table 1594 - E-switch Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	vport_svlan_strip	If set, svlan strip for received packets is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	
	30	vport_cvlan_strip	If set, cvlan strip for received packets is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	
	29	vport_svlan_insert	If set, svlan insertion for transmitted packets is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	

Table 1594 - E-switch Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
00h	28	vport_cvlan_insert_if_not_exist	If set, cvlan insertion for transmitted packets if it does not exist is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	
	27	vport_cvlan_insert_overwrite	If set, cvlan insertion for transmitted packets or overwrite when it does not exist is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	
	26	not_to_use	Don't use this field. was allocated for another field	
	25	vport_cvlan_insert_always	If set, cvlan insert_always mode in vport_cvlan_insert is supported in e-switch. See Section 8.6, “Esw_Vport Context - E-switch Virtual Port Context,” on page 284.	
	24	esw_shared_ingress_acl	If set, the device supports sharing an INGRESS ACL root flow table among multiple e-switch vports on same e-switch, by setting the table_of_other_vport and table_vport_num in SET_FLOW_TABLE_ROOT command.	
	23	esw_uplink_ingress_acl	If set, the device supports INGRESS ACL for the uplink vport.	
	22	root_ft_on_other_esw	If set, the device supports using a root flow table of another e-Switch by setting table_eswitch_owner_vhca_id_valid and table_eswitch_owner_vhca_id in SET_FLOW_TABLE_ROOT command. (Supported for FDB and Ingress Flow table only)	
	21	sw_network_metadata	If set, then sw_network_metadata in NIC vport context is supported.	
	7	match_uplink_other_esw	If set, then match on uplink of other e-switch is supported.	
	6	esw_functions_changed	If set, QUERY_ESW_FUNCTIONS command and ESW_FUNCTIONS_CHANGED event is supported.	
	5	fdb_single_tir_destination	If set, a single TIR can be used as a destination in FDB Flows.	
	4	ecpf_vport_exists	If set, the E-switch includes one more vport for Embedded CPU function (vport number 0xfffe).	
	3	counter_eswitch_affinity	Defines which eSwitch should allocate the Flow Counter attached to a Flow the FDB table, when the Flow matches the source_vport and the eswitch_id (using the merge_eswitch feature). 0x0: SOURCE_ESWITCH - The eSwitch matched in the eswitch_id. 0x1: FLOW_ESWITCH - The eSwitch to which the Flow is added	

Table 1594 - E-switch Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
00h	1	nic_vport_node_guid_modify	If set, node_guid modify is supported by vport_group_manager to other vports in nic vport_context. See Table 2507, “MODIFY_NIC_VPORT_CONTEXT Input Structure Layout,” on page 2164.	
	0	nic_vport_port_guid_modify	If set, port_guid modify is supported by vport_group_manager to other vports in nic vport_context. See Table 2507, “MODIFY_NIC_VPORT_CONTEXT Input Structure Layout,” on page 2164.	
04h	31	vxlan_encap_decap	If set, encapsulation and decapsulation of VXLAN header is supported.	
	30	nvgre_encap_decap	If set, encapsulation and decapsulation of NVGRE header is supported.	
	29	encap_general_header	If set, encapsulation of General header is supported. Supporting general header encapsulation does not necessarily mean that offload and parsing is also supported. So SW need to make sure the encapsulated protocol can be parsed add off-loaded is needed.	
	28:24	log_max_fdb_encap_uplink	Log (base 2) of the number of supported packet reformat Contexts for FDB Flow Table Type which can be created to be attached to an Uplink destination. Value 0x0 indicates this feature is not supported. See Section 8.18.3.6, “Allocating a Packet Reformat Context,” on page 422	
	20:16	log_max_packet_reformat_context	Log (base 2) of the maximum packet reformat contexts that can be created by ALLOC_PACKET_REFORMAT_CONTEXT.	
	9:0	max_encap_header_size	The maximum size of the encapsulation header supported by ALLOC_PACKET_REFORMAT_CONTEXT for NIC related flow tables	
08h	20:16	log_max_esw_sf	Log(base 2) of maximum of total number of Sub-functions supported on eswitch (for both host PF and ECPF).	
	15:0	esw_sf_base_id	Base sub-function identifier on eswitch that also define the vport number on eswitch.	

29.3.4.8 Vector Calc Capabilities

Querying for vector calc capabilities provides detailed vector calc operations capabilities. Before querying these capabilities, software must make sure that the vector_calc field in HCA_CAP is equal to 1.

Table 1595 - Vector Calc Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
calc matrix type 8bits																															00h	
calc matrix																																
									max_vector_count																		max_chunk_size		04h			
															calc_operation_0															08h		
															calc_operation_1															0Ch		
															calc_operation_2															10h		
															calc_operation_3															14h		
																															18h-FCh	

Table 1596 - Vector Calc Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	calc_matrix	If set, Calc matrix is supported.	
	30	calc_matrix_type_8bits	If set, then Galois multipliers type of 8bits is supported. See Table 77, “Vector Calc Segment Format,” on page 336.	
04h	23:16	max_vector_count	Maximum number of vectors supported.	
	2:0	max_chunk_size	Maximum chunk size supported.	

Table 1596 - Vector Calc Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
08h	15:0	calc_operation_0	Bit mask indicates which vector calc operations are supported by calc_operation_0 in vector calc segment. Bit 1: add operation supported. Bit 2: max operation supported Bit 3: and operation supported Bit 4: or operation supported Bit 5: xor operation supported Bit 6: min operation supported Bit 7: swap_endianness operation supported	
0Ch	15:0	calc_operation_1	Bit mask indicates which vector calc operations are supported by calc_operation_1 in vector calc segment. Bit 5: xor operation supported.	
10h	15:0	calc_operation_2	Bit mask indicates which vector calc operations are supported by calc_operation_2 in vector calc segment. Bit 5: xor operation supported.	
14h	15:0	calc_operation_3	Bit mask indicates which vector calc operations are supported by calc_operation_3 in vector calc segment. Bit 5: xor operation supported.	

29.3.4.9 Quality of Service Capabilities

Querying for quality of service capabilities provides detailed QoS capabilities. Before querying these capabilities, software must make sure that the HCA_CAP.qos==1.

Table 1597 - Quality of Service Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
				hll				packet_pacing_burst_bound	packet_pacing_typical_size	flow_meter_old	nic_sq_scheduling	nic_bw_share	nic_rate_limit	packet_pacing_uid	log_esw_max_sched_dept_h	log_max_flow_meter								flow_meter_reg_id												00h
				log_nic_max_sched_dept_h				flow_meter				log_max_qos_nic_queue_group									max_flow_meter_bs_exponent												04h			
packet_pacing_max_rate																																08h				

Table 1597 - Quality of Service Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
packet_pacing_min_rate																																0Ch
																packet_pacing_rate_table_size																10h
esw_element_type																esw_tsar_type																14h
																max_qos_para_vport																18h
max_tsar_bw_share																																1Ch
eswnic_element_type																nic_tsar_type																20h
																																24h-FCh

Table 1598 - Quality of Service Capabilities Field Descriptions

Offset	Bits	Name	Description	Access	Modifiable by vhca_resource_manager
00h	31	packet_pacing	If set, packet pacing is supported. See Section 29.36, “QoS Commands - Quality of Service Commands,” on page 2199.		YES [Trust]
	30	esw_scheduling	If set, E-Switch scheduler can be built to define QoS, see Section 19.1, “Transmit Scheduling,” on page 925		NO [FW]
	29	esw_bw_share	When set, eSwitch scheduler supports non-default BW allocation for scheduling elements.		NO [FW]
	28	esw_rate_limit	When set, eSwitch scheduler supports setting rate limit for scheduling elements.		NO [FW]
	27	hll	When set, the device supports Head of queue Lifetime Limit for Ethernet ports. See		NO [FW]
	26	packet_pacing_burst_bound	When set, setting an upper bound for packet bursts of a packet pacing rate limit is supported.		NO [Unlimited]
	25	packet_pacing_typical_size	When set, each rate can be associated with a typical packet size.		NO [Unlimited]

Table 1598 - Quality of Service Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access	Modifiable by vhca_resource_manager
00h	24	flow_meter_old	If set, the device supports Flow Meters. [Internal] Deprecated by flow_meter due to FW bug in reg C MSB usage. FW should still report it.		NO [FW]
	23	nic_sq_scheduling	When set, NIC Transmit scheduler can be built to define QoS between Send Queues (SQs).		YES [Trust]
	22	nic_bw_share	When set, NIC Transmit scheduler supports non-default BW allocation for scheduling elements. Valid only if nic_sq_scheduling is set.		NO [Unlimited]
	21	nic_rate_limit	When set, NIC Transmit scheduler supports setting rate limit for scheduling elements. Valid only if nic_sq_scheduling is set.		NO [Unlimited]
	20	packet_pacing_uid	If set, Packet pacing can be set with UID!=0.		No [FW]
	19:16	log_esw_max_sched_depth	Log (base 2) of the longest path supported from the root TSAR to the scheduler leaves. For example, value 0x0 indicates the leaves must be attached to the root TSAR.		NO [FW]
	15:8	log_max_flow_meter	Log (base 2) of the maximal number of supported Flow Meter Objects (actual number may be lower)		NO [FW]
	7:0	flow_meter_reg_id	Bit map per register_c ID indicating whether this register can be used for Flow Meter result. set bit means this register is supported. Bit 0: REG_C_0 Bit 1: REG_C_1 Bit 2: REG_C_2 Bit 3: REG_C_3 Bit 4: REG_C_4 Bit 5: REG_C_5 Bit 6: REG_C_6 Bit 7: REG_C_7		NO [FW]

Table 1598 - Quality of Service Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access	Modifiable by vhca_resource_manager
04h	27:24	log_nic_max_sched_depth	Log (base 2) of the longest path supported from the root TSAR to the NIC scheduler leaves (QUEUE_GROUPS). For example, value 0x0 indicates the leaves must be attached to the root TSAR. Valid only if nic_sq_scheduling is set.		YES [Trust]
	23	flow_meter	If set, the device supports Flow Meters.		No [FW]
	20:16	log_max_qos_nic_queue_group	Log (base 2) of the Maximal number of supported Scheduling elements of type QUEUE_GROUP for NIC Transmit Scheduling. Valid only if nic_sq_scheduling is set.		YES [Trust]
	7:0	max_flow_meter_bs_exponent	Maximal value supported for Bucket Size mantissa (CBS or EBS)		NO [FW]
08h	31:0	packet_pacing_max_rate	The maximal rate limit supported by Packet Pacing. Value is given in Kb/s.		NO [FW]
0Ch	31:0	packet_pacing_min_rate	The minimal rate limit supported by Packet Pacing. Value is given in Kb/s.		NO [FW]
10h	15:0	packet_pacing_rate_table_size	The size of packet pacing table.		NO [FW]
14h	31:16	esw_element_type	Bit mask indicates the supported Scheduling element types in E-Switch Scheduler. Set bit indicates the corresponding type is supported. Bit 0: TSAR Bit 1: VPORT Bit 2: VPORT_TC Bit 3: PARA_VPORT_TC		NO [FW]
	15:0	esw_tsar_type	Bit mask indicates the supported TSAR types in E-Switch Scheduler. Set bit indicates the corresponding type is supported. Bit 0: DWRR Bit 1: Round Robin Bit 2: ETS		NO [FW]
18h	15:0	max_qos_para_vport	Maximal number of Para-VPorts supported. Value 0x0 indicates Para-VPorts are not supported.		NO [FW]
1Ch	31:0	max_tsar_bw_share	Maximal value allowed for a TSAR agent BW share (<i>bw_share</i>)		NO [FW]

Table 1598 - Quality of Service Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access	Modifiable by vhca_resource_manager
20h	31:16	nic_element_type	Bit mask indicates the supported Scheduling element types in NIC Scheduler. Set bit indicates the corresponding type is supported. Bit 0: TSAR Bit 1: VPORT - [Internal] Bit 2: VPORT_TC - [Internal] Bit 3: PARA_VPORT_TC - [Internal] Bit 4: QUEUE_GROUP		NO [FW]
	15:0	nic_tsar_type	Bit mask indicates the supported TSAR types in NIC Scheduler. Set bit indicates the corresponding type is supported. Bit 0: DWRR Bit 1: Round Robin Bit 2: ETS		NO [FW]

29.3.4.10 Debug Capabilities

Querying for debug capabilities provides detailed capabilities. Before querying these capabilities, software must make sure that the HCA_CAP.debug==1. See Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1599 - Debug Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
core_dump_gp				log_cr_dump_to_mem_size					resource_dump		log_min_resource_dump_eq														log_max_samples							00h
core_dump_general		health_mon_rx_activity	diag_counter_tracer_dump																													
		repetitive																														
		single																														

Table 1599 - Debug Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-3Ch
diagnostic_counter[0] (See Table 1601, “Diagnostic_Counter Layout,” on page 1779)																																40h
diagnostic_counter[1] (See Table 1601, “Diagnostic_Counter Layout,” on page 1779)																																44h
...																																48h-..

Table 1600 - Debug Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	core_dump_general	If set, Core dump of type “General” is supported.	
	30	core_dump_qp	If set, Core dump of type of specific QP is supported.	
	27:23	log_cr_dump_to_mem_size	Log(base 2) of the size in granularity of 4KB to be allocated by host in order to accommodate cr_dump. 0 means feature is not supported. See Table 935, “CORE_DUMP Register Layout,” on page 1128	
	22	resource_dump	If set, Resource_dump register is supported. See Table 937, “RESOURCE_DUMP Register Layout,” on page 1130	
	20:16	log_min_resource_dump_eq	Log(base 2) of the the recommended minimal size of eq to handle the resource_dump_event	
	7:0	log_max_samples	The maximum number of samples that can be stored on the device’s sampling buffer is $2^{\text{log\_max\_samples}}$ in counters unit (i.e. 100 will represent the ability to store 100 samples of single counter, 50 samples of 2 counters, etc).	

Table 1600 - Debug Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	31	single	Single sampling mode is supported	
	30	repetitive	Repetitive sampling mode is supported	
	29	health_mon_rx- _activity	If set, health monitoring for rx path activity is supported. See Section 26.5.1, “RX Path Activity,” on page 988	
	28	diag_counter_trace r_dump	If set, the device supports dumping the diagnostic counters reports into the tracer buffer.	
	7:0	log_min_sam- ple_period	The minimal sample period is $2^{\text{log\_min\_smample\_period}}$ in device clocks. Clock rate of the device is reported by HCA_ - CAP.device_frequency_khz.	
40h-..	31:0	diagnos- tic_counter[...] (See Table 1601, “Diagnos- tic_Counter Lay- out,” on page 1779)	List of counters supported. Number of counters reported by num_of_counters.	

Table 1601 - Diagnostic_Counter Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sync																counter_id																00h

Table 1602 - Diagnostic_Counter Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	sync	The counter might be used as Synced Start Diagnostics Count- ers	
	15:0	counter_id	Diagnostic counter identifier.	

29.3.4.11 NVMe-oF Capabilities

Querying for NVMe-oF capabilities provides detailed capabilities on NVMe-oF. Before querying these capabilities, software must make sure that the HCA_CAP.nvmf_target_offload==1. See Table 1571, “HCA Capabilities Layout,” on page 1671).

Please refer to Section 20.5, “NVM Express over Fabric (NVMe-oF) Target Application Offload,” on page 999 for further details.

Table 1603 - NVMe-oF Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
release_xrq_error				log_max_frontend_nsid						log_min_cmd_timeout						log_max_cmd_timeout						log_max_namespace_per_xrq								00h			
						read_write_flush_offload_dc		read_write_offload_dc		write_offload_dc		log_max_staging_buffer_size				log_min_staging_buffer_size				cmd_unknown_namespace_cnt		frontend_namespace_context		read_write_flush_offload_rc		read_write_offload_rc		read_offload_rc		write_offload_rc		04h	
								log_max_io_size				log_max_backend_controller				log_max_backend_controller_per_xrq				max_icdoff								08h					
								max_lba_metadata_size										log_max_queue_size						log_min_queue_size						0Ch			
								min_ioccsz																									
max_ioccsz																																14h	
nvmf_error_filters																						lossless_error_report		nvmf_error_reporting		nvmf_error_message_size				18h			
																						log_max_metadata_size						1Ch					
min_ioecsz																																+20h	
max_ioecsz																																+24h	

Table 1603 - NVMe-oF Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																4E20h-FCh

Table 1604 - NVMe-oF Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	release_xrq_error	If set, RELEASE_XRQ_ERROR command is supported. See Section 29.27.8, on page 2128	
	30	<u>nvmf_stats_report_back</u>	<u>If set, then report back XRQ statistics is supported.</u> <u>i.e. nvmf_stats_report_back_en in XRQ context can be set.</u>	
	28:24	log_max_frontend_nsid	Log (base 2) of maximum frontend NSID that can be used when attaching namespaces.	
	20:16	log_min_cmd_timeout	Log(base2) of the minimum supported command timeout (in Useconds) in NVMF_BACKEND_CONTROLLER_CONTEXT. See Table 2423, “NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130	
	12:8	log_max_cmd_timeout	Log(base2) of the maximum supported command timeout(in Useconds) in NVMF_BACKEND_CONTROLLER_CONTEXT. 0 means the device sets a default value. See Table 2423, “NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130	
	4:0	log_max_namespace_per_xrq	Log (base2) of the maximum number of namespaces supported by XRQ. Must be less than HCA_CAP.log_max_namespace_per_xrq	

Table 1604 - NVMe-oF Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	25	read_write_flush_offload_dc	If set, NVMe-oF offload of types read, write and flush for DC transport service are supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	24	read_write_offload_dc	If set, NVMe-oF offload of types read and write for DC transport service are supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	23	read_offload_dc	If set, NVMe-oF offload of type read for DC transport service is supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	22	write_offload_dc	If set, NVMe-oF offload of type write for DC transport service is supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	20:16	log_max_staging_buffer_size	The maximum size of NVMe-oF staging buffer in bytes is $2^{(\log\_max\_staging\_buffer\_size + nvme\_memory\_log\_page\_size)}$.	
	12:8	log_min_staging_buffer_size	The minimum size of NVMe-oF staging buffer in bytes is $2^{(\log\_max\_staging\_buffer\_size + nvme\_memory\_log\_page\_size)}$.	

Table 1604 - NVMe-oF Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	6	cmd_unknown_namespace_cnt	If set, cmd_unknown_namespace_cnt in NVMe-oF Target Offload Context is supported. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519	
	5	frontend_namespace_context	If set, QUERY_NVMMF_NAMESPACE_CONTEXT command is supported.	
	4	read_write_flush_offload_rc	If set, NVMe-oF offload of types read, write and flush for RC transport service are supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	3	read_write_offload_rc	If set, NVMe-oF offload of types read and write for RC transport service are supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	2	read_offload_rc	If set, NVMe-oF offload of type read for RC transport service is supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
	1	write_offload_rc	If set, NVMe-oF offload of type write for RC transport service is supported. See Table 257, “NVMe-oF Target Offload Context Field Descriptions,” on page 520	
08h	28:24	log_max_io_size	Log (base 2) of the maximum supported nvmmf io size in granularity of bytes.	
	20:16	log_max_backend_controller	Log (base 2) of the maximum supported backend controllers.	
	12:8	log_max_backend_controller_per_xrq	Log (base 2) of the maximum supported backend controllers per XRQ.	
	7:0	max_icdoff	The maximum supported icdoff in NVMe-oF target offload context header. See Table 256, “NVMe-oF Target Offload Context Header Layout,” on page 519.	

Table 1604 - NVMe-oF Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
0Ch	31:16	max_lba_meta-data_size	The maximum supported LBA metadata size that can be set in ATTACH_NVMMF_NAMESPACE. See Table 2433, “ATTACH_NVMMF_NAMESPACE Input Structure Layout,” on page 2135	
	12:8	log_max_queue_size	Log (base 2) of the maximum supported nvme queue size in granularity of bytes.	
	4:0	log_min_queue_size	Log (base 2) of the minimum supported nvme queue size in granularity of bytes.	
10h	31:0	min_ioccsz	The minimum supported iocc size in granularity of 16bytes	
14h	31:0	max_ioccsz	The maximum supported iocc size in granularity of 16bytes	
18h	31:16	nvmmf_error_filters	Bitmask indicates the supported filters for NVMMF error reporting by XRQ. To enable specific filters, SW should set nvmmf_error_filters in XRQ Context. Bit 0: report_CC_check_originated_errors Bit 1: report_disk_originated_errors_with_more_bit Bit 2: report_disk_originated_errors_without_more_bit Bit 3: report_good_disk_completions - (for debug)	
	5	lossless_error_report	If set, lossless error reporting is supported by the NVMMf target offload engine. i.e. lossless_error_report can be set in XRQ Context.	
	4	nvmmf_error_reporting	If set, NVMMf target offload error reporting is supported. i.e. error_report_qpn can be set in XRQ Context.	
	3:0	nvmmf_error_message_size	Size of NVMMf Error Message in units of 64B, See Table 258, “NVMMf Error Message Layout,” on page 524	
1Ch	4:0	log_max_metadata_size	Log(base 2) of the the maximum supported metadata size per nvmmf io transaction.	
20h	31:0	min_iocsz	Minimal allowed value for iocsz in 16B granularity. 0 same as 1 (i.e.16B).	
24h	31:0	max_iocsz	Maximal allowed value for iocsz in 16B granularity. 0 same as 1 (i.e.16B).	

29.3.4.12 Device Memory Capabilities

Querying for Device Memory capabilities provides detailed capabilities for the device internal memory. Before querying these capabilities, software must make sure that the HCA_CAP.device_memory==1. See Table 1571, “HCA Capabilities Layout,” on page 1671).

Please refer to Section 18.4, “MEMIC - Memory Mapped to InterConnect,” on page 923 for further details.

Table 1605 - Device Memory Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
memic																														00h		
										log_min_memic_alloc_size													log_max_memic_ad- dr_alignment					04h				
memic_bar_start_addr																													08h-0Ch			
memic_bar_size																													10h			
max_memic_size																													14h			
steering_sw_icm_start_address																													18h-1Ch			
log_header_modify_argu- ment_sw_icm_size								log_header_modi- fy_sw_icm_size										log_sw_icm_al- loc_granularity					log_steering_sw_icm_size						20h			
																								log_header_modify_pat- tern_sw_icm_size					24h			
header_modify_sw_icm_start_address																													28h-2Ch			
header_modify_argument_sw_icm_start_address																													30h-34h			
header_modify_pattern_sw_icm_start_address																													38h-3Ch			
																													40h-FCFh			

Table 1606 - Device Memory Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	memic	If set, MEMIC is supported. See Section 18.4, “MEMIC - Memory Mapped to InterConnect,” on page 923.	

Table 1606 - Device Memory Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	20:16	log_min_memic_alloc_size	Log (base 2) of the minimum allocation size of MEMIC in granularity of bytes.	
	7:0	log_max_memic_addr_alignment	The maximum alignment of MEMIC address that can be required in MEMIC allocation is $64 * 2^{\text{log\_max\_memic\_addr\_alignment}}$	
08h-0Ch	64	memic_bar_start_addr	Start address of MEMIC area on the BAR.	
10h	31:0	memic_bar_size	Size of MEMIC area on the BAR starting from memic_bar_start_addr in a granularity of bytes.	
14h	31:0	max_memic_size	Maximum memic size supported in bytes.	
18h-1Ch	64	steering_sw_icm_start_address	ICM base address reserved for SW managed steering, given in Bytes	
20h	31:24	log_header_modify_argument_sw_icm_size	Log (base 2) of the ICM range reserved for SW managed Header Modify Argument contexts, given in bytes.	
	23:16	log_header_modify_sw_icm_size	Log (base 2) of the ICM range reserved for SW managed Header Modify contexts, given in bytes.	
	13:8	log_sw_icm_alloc_granularity	Log (base 2) of the minimum allocation granularity of SW ICM in granularity of bytes.	
	7:0	log_steering_sw_icm_size	Log (base 2) of the ICM range reserved for SW managed steering, given in bytes.	
24h	7:0	log_header_modify_pattern_sw_icm_size	Log (base 2) of the ICM range reserved for SW managed Header Modify Pattern contexts, given in bytes.	
28h-2Ch	64	header_modify_sw_icm_start_address	ICM base address reserved for SW managed Header Modify contexts, given in Bytes	
30h-34h	64	header_modify_argument_sw_icm_start_address	ICM base address reserved for SW managed Header Modify Arguments contexts, given in Bytes	
38h-3Ch	64	header_modify_pattern_sw_icm_start_address	ICM base address reserved for SW managed Header Modify Pattern contexts, given in Bytes	

29.3.4.13 NVME Device Emulation Capabilities

Querying for Device Emulation capabilities provides detailed capabilities of the NVME device emulation for the NVME device emulation manager. Before querying for these capabilities, software must make sure that the HCA_CAP.nvme_device_emulation_manager==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1607 - NVME Device Emulation Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset									
																																00h									
nvme offload type SOE		nvme offload type doorbell only		nvme offload type command capsule		log_max_nvme_offload_namespaces																	max_emulated_devices																		
																registers_size													04h												
																		log_max_emulated_sq						log_max_emulated_cq					08h												
																																0Ch-FCh									

Table 1608 - NVME Device Emulation Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	nvme_offload_type_SQE	If set, offload type of SQE is supported in Device Emulation QP to NVMe MAP Object, See Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542.	
	30	nvme_offload_type_doorbell_only	If set, offload type of DOORBELL_ONLY is supported in Device Emulation QP to NVMe MAP Object, See Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542.	
	29	nvme_offload_type_command_capsule	If set, offload type of Command Capsule is supported in Device Emulation QP to NVMe MAP Object, See Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542.	
	28:24	log_max_nvme_offload_namespaces	Log(base 2) of the maximum number of namespaces supported per NVME SQ in Device Emulation QP to NVMe MAP Object, See Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542.	
	7:0	max_emulated_devices	Total number of emulated devices managed by the device emulation manager (physical functions devices). Information about these Devices can be reported by QUERY_EMULATED_FUNCTIONS_INFO. See Table 1962, “QUERY_EMULATED_FUNCTIONS_INFO Input Structure Layout,” on page 1940.	
04h	15:0	registers_size	Size of device emulation registers	
08h	12:8	log_max_emulated_sq	Log(base 2) of the maximum supported emulated SQs.	
	4:0	log_max_emulated_cq	Log(base 2) of the maximum supported emulated CQs.	

29.3.4.14 TLS Capabilities

Querying for TLS capabilities provides detailed capabilities of the TLS offload. Before querying for these capabilities, software must make sure that the HCA_CAP.tls_tx/tls_rx==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1609 - TLS Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
tls_1_2_aes_gcm_128	tls_1_3_aes_gcm_128	tls_1_2_aes_gcm_256	tls_1_3_aes_gcm_256																													00h
																																04h-FCh

Table 1610 - TLS Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	tls_1_2_aes_gcm_128	If set, indicates support of the aes_gcm cipher with TLS1.2 and 128 bit key	
	30	tls_1_3_aes_gcm_128	If set, indicates support of the aes_gcm cipher with TLS1.3 and 128 bit key	
	29	tls_1_2_aes_gcm_256	If set, indicates support of the aes_gcm cipher with TLS1.2 and 256 bit key	
	28	tls_1_3_aes_gcm_256	If set, indicates support of the aes_gcm cipher with TLS1.3 and 256 bit key	

29.3.4.15IPSEC Capabilities

Querying for IPsec Offload capabilities provides detailed capabilities and types of supported IPsec offloads, their limitations and variants.

Before querying for these capabilities, software must make sure that the HCA_CAP.ipsec_offload is set (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1611 - IPSEC Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
				ipsec crypto esp aes gcm 128 decrypt								log_max-ipsec_offload																				00h				
				ipsec crypto esp aes gcm 256 decrypt																																
				ipsec crypto esp aes gcm 128 encrypt																																
				ipsec crypto esp aes gcm 256 encrypt																																
				ipsec esn																																
				ipsec crypto offload																																
				ipsec full offload																																
min_log_ipsec_full_replay_window								max_log_ipsec_full_replay_window																								04h				
																																				08h-FCh

Table 1612 - IPSEC Capabilities Field Descriptions

Offset	Bits	Name	Description	Access	Modifiable by vhca_resource_manager
00h	31	ipsec_full_offload			Yes [Trust]
	30	ipsec_crypto_offload			Yes [Trust]
	29	ipsec_esn			
	28	ipsec_crypto_esp_aes_gcm_256_encrypt			
	27	ipsec_crypto_esp_aes_gcm_128_encrypt			
	26	ipsec_crypto_esp_aes_gcm_256_decrypt			
	25	ipsec_crypto_esp_aes_gcm_128_decrypt			
	20:16	log_max_ipsec_offload	Log (base 2) of the maximum IPsec Offload contexts supported		
04h	31:24	min_log_ipsec_full_replay_window			
	23:16	max_log_ipsec_full_replay_window			

29.3.4.16 DPP Capabilities

Querying for DPP capabilities provides detailed capabilities of the DPP offload. Before querying for these capabilities, software must make sure that the `HCA_CAP.dpp==1` (see Table 1571, “HCA Capabilities Layout,” on page 1671).

More details on DPP, please refer to TBD

Table 1613 - DPP Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
dpp_wire_protocol																															00h-04h	
max_scatter_size																	log_min_dp- p_segment_size				log_max_dp- p_segment_size		08h									
																			log_max_dp- p_buffer_size		max_psn_size_supported					0Ch						
																															10h-FCh	

Table 1614 - DPP Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	dpp_wire_protocol	Bit mask indicates which DPP wire protocols are supported.	
08h	31:16	max_scatter_size	Maximum	
	12:8	log_min_dpp_seg- ment_size	Log(base 2) of the minimum supported size of scattered data per packet in granularity of Bytes.	
	4:0	log_max_dpp_seg- ment_size	Log(base 2) of the maximum supported size of scattered data per packet in granularity of Bytes.	
0Ch	12:8	log_max_dp- p_buffer_size	Log(base 2) of the maximum supported buffer size of in granu- larity of Bytes.	
	7:0	max_psn_size_ supported	The maximum number of bits in packet to represent PSN.	

29.3.4.17 VIRTIO Emulation Capabilities

Querying for VIRTIO Emulation capabilities provides detailed capabilities of the VDPA/VirtIO_NET and VIRTIO_BLK Emulation offload (depending on the opmode in QUERY_HCA_CAP).

Before querying for these capabilities, software must make sure that:

For virtio_net emulation: HCA_CAP.virtio_net_device_emulation_manager ==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

For virtio_blk emulation: `HCA_CAP.virtio_blk_device_emulation_manager == 1` (see Table 1571, “HCA Capabilities Layout,” on page 1671).

For VDPA emulation: `HCA_CAP.general_obj_types.VIRTQ_NET_Q == 1` (see Table 1571, “HCA Capabilities Layout,” on page 1671).

For further details on VIRTIO Emulation, please refer to Section 20.7, “VIRTIO Emulation Offload,” on page 1008.

Table 1615 - VIRTIO_Emulation Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																	
desc_tunnel_offload_type			eth_frame_offload_type			virtio_version_1_0			tso_ipv4			tso_ipv6			tx_csum			rx_csum																		event_mode			virtio_queue_type										00h
max_tunnel_desc																	log_doorbell_stride					log_doorbell_bar_size					04h																						
doorbell_bar_offset																															08h-0Ch																		
max_emulated_devices									max_num_virtio_queues																						10h																		
																															14h-1Ch																		
umem_1_buffer_param_a																															20h																		
umem_1_buffer_param_b																															24h																		
umem_2_buffer_param_a																															28h																		
umem_2_buffer_param_b																															2Ch																		
umem_3_buffer_param_a																															30h																		
umem_3_buffer_param_b																															34h																		
																															438h-FCh																		

Table 1616 - VIRTIO_Emulation Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	desc_tunnel_offload_type	When set descriptor tunneling is supported. See Virtio emulation offload section.	
	30	eth_frame_offload_type	When set descriptor converting virtio-net desc to ethernet frames is supported. See Virtio emulation offload section.	
	29	virtio_version_1_0	If set device is virtio 1.0 compliant	
	28	tso_ipv4	If set device supports virtio TSO for IPV4 packets	
	27	tso_ipv6	If set device supports virtio TSO for IPV6 packets	
	26	tx_csum	If set device supports virtio csum offload for Tx packets	
	25	rx_csum	If set device supports virtio csum offload for Rx packets	
	15:8	event_mode	Bit mask indicates which event modes are supported in VIRTIO Q. Bit 0x0: NO_MSIX_MODE Bit 0x1: QP_MODE Bit 0x2: MSIX_MODE	
	7:0	virtio_queue_type	Bitmask indicates the supported Virtio queue types supported. Bit 0: SPLIT Bit 1: PACKED	
04h	31:16	max_tunnel_desc	The maximum number of descriptors the device can tunnel on a single SEND operation. Valid only when hca_cap.desc_tunnel_offload_type is set.	
	12:8	log_doorbell_stride	Log(base 2) of stride between doorbells in granularity of Bytes. Reserved when QUERY_HCA_CAP.opmode is "Virtio_net_emulation_capabilities" or "Virtio_blk_emulation_capabilities"	
	4:0	log_doorbell_bar_size	Log(base 2) of doorbell Bar size in granularity of 4KB. Reserved when QUERY_HCA_CAP.opmode is "Virtio_net_emulation_capabilities" or "Virtio_blk_emulation_capabilities"	
08h-0Ch	64	doorbell_bar_offset	Start doorbell offset. Reserved when QUERY_HCA_CAP.opmode is "Virtio_net_emulation_capabilities" or "Virtio_blk_emulation_capabilities"	

Table 1616 - VIRTIO_Emulation Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
10h	31:24	max_emulated_devices	Max number of emulated devices that can be managed by the device emulation manager (physical functions devices). Information about these devices can be reported by QUERY_EMULATED_FUNCTIONS_INFO.	
	23:0	max_num_virtio_queues	maximum supported virtio queues.	
20h	31:0	umem_1_buffer_param_a	The size of UMEM 1 that need to provide to device when creating VIRTIO_Q is: buffer size = umem_1_buffer_param_a * virtio_q size + umem_1_buffer_param_b	
24h	31:0	umem_1_buffer_param_b		
28h	31:0	umem_2_buffer_param_a	The size of UMEM 2 that need to provide to device when creating VIRTIO_Q is: buffer size = umem_2_buffer_param_a * virtio_q size + umem_2_buffer_param_b	
2Ch	31:0	umem_2_buffer_param_b		
30h	31:0	umem_3_buffer_param_a	The size of UMEM 1 that need to provide to device when creating VIRTIO_Q is: buffer size = umem_3_buffer_param_a * virtio_q size + umem_3_buffer_param_b	
34h	31:0	umem_3_buffer_param_b		

29.3.4.18Event Capabilities

Querying for Event capabilities provides detailed capabilities and types of supported events. Before querying for these capabilities, software must make sure that the HCA_CAP.general_obj_types.event_cap ==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1617 - Event Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
user_affiliated_events_63_0																																00h-04h

Table 1617 - Event Capabilities Layout (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
user_affiliated_events_127_64																																08h-0Ch
user_affiliated_events_191_128																																10h-14h
user_affiliated_events_255_192																																18h-1Ch
user_unaffiliated_events_63_0																																20h-24h
user_unaffiliated_events_127_64																																28h-2CCh
user_unaffiliated_events_191_128																																30h-34h
user_unaffiliated_events_255_192																																38h-3CCh
																																40h-FCh

Table 1618 - Event Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	user_affiliated_events_63_0	Event Bitmask indicates the supported affiliated events. i.e. bit x indicates event_type=x is supported and it's affiliated with an object. For affiliated events, the first 8 Bytes has a generic format that report object_type and object_id (See Table 184, "Affiliated Event Header Layout," on page 482). Bit 0x27: OBJECT_CHANGE_EVENT Event Bitmask indicates the supported affiliated events. i.e. bit x indicates event_type=x is supported and it's affiliated with an object. For affiliated events, the first 8 Bytes has a generic format that report object_type and object_id (See Table 184, "Affiliated Event Header Layout," on page 482). Bit 0x1:Path_Migrated_Succeeded Bit 0x02:Communication_Established Bit 0x03:Send_Queue_Drained Bit 0x04:CQ_Error Bit 0x05:Local_WQ_Catastrophic_Error Bit 0x07:Path_Migration_Failed Bit 0x10:Invalid_Request_Local_WQ_Error Bit 0x11:Local_Access_Violation_WQ_Error Bit 0x12:Local_SRQ_Catastrophic_Error Bit 0x13>Last_WQE_Reached Bit 0x14:SRQ_Limit Bit 0x18:XRQ_ERROR Bit 0x1C:DCT_All_Connections_Closed Bit 0x1D: DCT_Access_Key_Violation Bit 0x27: OBJECT_CHANGE_EVENT	
08h-0Ch	64	user_affiliated_events_127_64	-	
10h-14h	64	user_affiliated_events_191_128	-	
18h-1Ch	64	user_affiliated_events_255_192	-	

Table 1618 - Event Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
20h-24h	64	user_unaffiliated_events_63_0	-Event Bitmask indicates the supported user unaffiliated events. i.e. bit x indicates event_type=x is supported and unaffiliated. Bit 0x09: Port_State_Change Bit 0x0D: Nic_Vport_Change Bit 0x0E: ESW_FUNCTIONS_CHANGED Bit 0x16: Port_Module_Event Bit 0x17: Temp_Warning_Event Bit 0x25: PPS_Event Bit 0x22: General_Notification_Event	
28h-2Ch	64	user_unaffiliated_events_127_64		
30h-34h	64	user_unaffiliated_events_191_128		
38h-3Ch	64	user_unaffiliated_events_255_192		

29.3.4.19 HotPlug Capabilities

Querying for HotPlug capabilities provides detailed capabilities hotplug. Before querying for these capabilities, software must make sure that the HCA_CAP.hotplug_manager ==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1619 - Hotplug Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hotplug_device_types_supported																log_max_bar_size								max_hotplug_devices								00h
																																04h-FCch

Table 1620 - Hotplug Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	hotplug_device_types_supported	Bit mask indicates the supported hotplug device types Bit 0: NVME Bit 1: VIRTIO_NET Bit 2: VIRTIO_BLK	
	15:8	log_max_bar_size	Log (base 2) of the maximum bar size for a hotplug device.	
	7:0	max_hotplug_devices	Maximum number of devices that can be hotplugged.	

29.3.4.20 Wqe Based Flow Table Capabilities

Querying for Wqe Based Flow table capabilities provides detailed capabilities of Wqe Based Flow Table feature. Before querying for these capabilities, software must make sure that the HCA_CAP.wqe_based_flow_table_update ==1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1621 - WQE_BASED_FLOW_TABLE Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
wqe_based_flow_update_actions_supported																																00h-04h
																																08h-FCh

Table 1622 - WQE_BASED_FLOW_TABLE Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	wqe_based_flow_update_actions_supported	Bitmap per Action ID, when set this action is supported for Flow Table Entry definition: 0: NOP 1: Transmit 2: inline_qpn 3: qp_list 4: Iterator 5: copy 6: set 7: add 8: remove_by_size 9: remove_header2header 10: insert_with_inline 11: insert_with_pointer 12: flow_tag 13: Queue_ID_selection 14: accelerated_modify_action_list 15: modify_action_list 16: IPSec_Encryption 17: IPSec_Decryption 18: ASO 19: trailer 20: counter_id 21: extended_qpn 22: port_selection 23: count_on_source_gvmi 24: TLS 25: Clear	

29.3.4.21 NVMeoTCP Capabilities

Querying for NVMeTCP capabilities provides detailed capabilities of NVMeoTCP offload feature. Before querying for these capabilities, software must make sure that the `HCA_CAP.nvmeotcp == 1` (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1623 - NVMeoTCP Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
zerocopy			crc_tx		crc_rx														log_max_ queue_depth						version						00h	
																															04h-FCh	

Table 1624 - NVMeoTCP Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	zerocopy	If set, zero copy is supported.	
00h	30	crc_rx	If set, CRC32 for received data is supported.	
00h	29	crc_tx	If set, CRC32 for transmitted data is supported.	
00h	12:8	log_max_- queue_depth	Log(base 2) of the maximum supported NVMe queue depth	
00h	7:0	version	Bitmask indicates the supported NVMeoTCP versions as reported in ICresp. Bit0: version_0	

29.3.4.22 Crypto Capabilities

Querying for Crypto capabilities provides detailed capabilities of Cryptographic offload feature. Before querying for these capabilities, software must make sure that the HCA_CAP.crypto == 1 (see Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 1625 - Crypto Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset			
wrapped_crypto_operational																										wrapped_import_method								00h	
														log_max_num_deks								log_max_num_import_keks								log_max_num_creds				04h	
failed_selftests																num_nv_import_keks								num_nv_credentials								08h			
																																0Ch-FCh			

Table 1626 - Crypto Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	wrapped_crypto_operational	If set to 1, it indicates that the crypto engines in wrapped import method are operational. If set to 0, SW should change it to operational mode. See Table 943, “CRYPTO_OPERATIONAL Register Layout,” on page 1135.	
	7:0	wrapped_import_method	Bitmap indicating for each engine/offload if it is in Plaintext or Wrapped mode. If set, the mode is Wrapped. 0: TLS - If set, TLS is in wrapped mode 1: IPSEC - If set, IPSEC is in wrapped mode 2: AES-XTS - If set, AES-XTS is in wrapped mode	

Table 1626 - Crypto Capabilities Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	20:16	log_max_num_deks	Log (base 2) of maximal number of DEKs supported [Internal] This replaces the one in HCA_CAP.	
	12:8	log_max_num_import_keks	Log (base 2) of maximal number of Imported KEKs supported	
	4:0	log_max_num_creds	Log (base 2) of maximal number of Credentials supported	
08h	31:16	failed_selftests	Bitmap for selftests. If set, the respective test has failed. Depending on which tests have failed, the respective crypto capability may be turned off as a result 0x1: test_aes_gcm- AES GCM test 0x2: test_aes_ecb- AES ECB test 0x4: test_aes_xts- AES XTS test 0x8: test_hmac_sha- HMAC SHA test 0x10: test_sha- SHA test	
	15:8	num_nv_import_keks	<u>Number of non-volatile import_KEK slots out of the total import KEKs</u>	
	7:0	num_nv_credentials	<u>Number of non-volatile credential slots out of the total credentials</u>	

29.3.5 SET_HCA_CAP – Set Device Capabilities

The SET_HCA_CAP command defines the device limitations and capabilities supported. Since modifying some capabilities affects resources required during INIT_HCA, this command must be run prior to initial QUERY_PAGES command.

Table 1627 - SET_HCA_CAP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
other_function																function_id												08h				

Table 1627 - SET_HCA_CAP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch
capability																																10h-100Ch

Table 1628 - SET_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Bit[0]: Reserved. Bits[15:1] Indicates Capability Type 0x0: General Device Capabilities 0x1: Ethernet Offload Capabilities 0x2: ODP Capabilities 0x3: Atomic Capabilities 0x4: RoCE Capabilities 0x5: IPoIB Basic Offload Capabilities 0x6: IPoIB Enhanced Offload Capabilities 0x7: NIC Flow Table 0x8: E-switch Flow Table Capabilities 0x9: E-switch Capabilities 0xA: Snapshot Capabilities 0xB: Vector Calc Capabilities 0xC: Quality of Service Capabilities 0xD: Debug Capabilities 0xE: NVMe-oF Capabilities 0xF: Device Memory Capabilities 0x10: NVME Device Emulation Capabilities 0x11: TLS Capabilities 0x12: DPP Capabilities 0x13: VDPA_EMULATION Capabilities 0x14: EVENT Capabilities 0x15: IPSEC Capabilities 0x16: VIRTIO_NET_EMULATION_Capabilities 0x17: VIRTIO_BLK_EMULATION_Capabilities 0x18: HOTPLUG_Capabilities 0x19: NVMEOTCP_Capabilities 0x1A: CRYPTO_Capabilities 0x1B: WOQ_BASED_FLOW_TABLE_Capabilities	

Table 1628 - SET_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	other_function	If set, indicates accessing capabilities of other function identified by function_id. Valid only for vhca resource manager (HCA_CAP.vhca_resource_manager==1)	
	15:0	function_id	function_id of vHCA: 0: When accessing my vHCA. 0xffff: Access all function_ids. others: Function_id of specific vHCA used by vhca_resource_manager (HCA_CAP.vhca_resource_manager==1).	

Table 1628 - SET_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-100Ch	32768	capability	Capability structure according to the op_mod. op_mod[0:0] Indicates Maximum or Current capabilities 0x0: Maximum 0x1: Current op_mod[15:1] Indicates Capability Type 0x0: General Device Capabilities. See Table 1571, “HCA Capabilities Layout,” on page 1671. 0x1: Ethernet Offload Capabilities. 0x2: ODP Capabilities. See Table 1573, “ODP Capabilities Layout,” on page 1726. 0x3: Atomic Capabilities. See Table 1577, “Atomic Capabilities Layout,” on page 1728. 0x4: RoCE Capabilities. See Table 1579, “RoCE Capabilities Layout,” on page 1732. 0x5: IPoIB Basic Offload Capabilities. See Table 1581, “Per Protocol Networking Offload Capabilities Layout,” on page 1734. 0x6: IPoIB Enhanced Offload Capabilities. <u>May:TBD:need to re-add cross rerefrence to Per Protocol Networking Offload Capabilities Layout table</u> 0x7: Nic Flow Table Capabilities. See Table 1583, “Flow Table NIC Capabilities Layout,” on page 1742 0x8: E-switch Flow Table Capabilities. See Table 1585, “Flow Table E-switch Capabilities Layout,” on page 1748 0x9: E-switch Capabilities. See Table 1593, “E-switch Capabilities Layout,” on page 1768 0xB: Vector Calc Capabilities. See Table 1595, “Vector Calc Capabilities Layout,” on page 1772 0xC: Quality of Service Capabilities. See Table 1597, “Quality of Service Capabilities Layout,” on page 1773 0xD: Debug Capabilities. See Table 1599, “Debug Capabilities Layout,” on page 1777 0xE: NVMe-oF Capabilities. See Table 1603, “NVMe-oF Capabilities Layout,” on page 1780 0xF: Device_Memory Capabilities. See Table 1605, “Device Memory Capabilities Layout,” on page 1785 0x10: Device Emulation Capabilities. See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787 0x11: TLS Capabilities. See Table 1609, “TLS Capabilities Layout,” on page 1789 0x12: DPP Capabilities. See Table 1613, “DPP Capabilities Layout,” on page 1792 0x13: VIRTIO_EMULATION Capabilities. See Table 1615, “VIRTIO_Emulation Capabilities Layout,” on page 1793 0x14: EVENT_Capabilities. See See Table 1617, “Event Capabilities Layout,” on page 1795 0x15: IPSEC Capabilities. See Table 1611, “IPSEC Capabilities Layout,” on page 1790 0x16: VIRTIO_NET_EMULATION_Capabilities 0x17: VIRTIO_BLK_EMULATION_Capabilities 0x18: HOTPLUG_Capabilities 0x19: NVMEOTCP_Capabilities 0x1A: CRYPTO_Capabilities 0x1B: WQE_BASED_FLOW_TABLE_Capabilities	

Table 1629 - SET_HCA_CAP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch

Table 1630 - SET_HCA_CAP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.6 QUERY_OTHER_HCA_CAP - Query Other HCA Capabilities

vHCA group manager (HCA_CAP.vhca_group_manager) is capable to control some capabilities in HCA_CAP (see Table 1571, “HCA Capabilities Layout,” on page 1671) of other vHCAs that are under its responsibility (for example, PF i responsible on VFs).

Note that configuring the capabilities of other vHCA using MODIFY_OTHER_HCA_CAP command must be done before enabling this vHCA by the vHCA group manager, and only after vHCA group manager perform INIT_HCA command.

Table 1631 - QUERY_OTHER_HCA_CAP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																function_id																08h
																																0Ch

Table 1632 - QUERY_OTHER_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	15:0	function_id	function_id of HCA: 0: When accessing my HCA. 0xffff: Access all function_ids. others: Function_id of specific vHCA used by vhca_group_manager (HCA_CAP.vhca_group_manager ==1).	

Table 1633 - QUERY_OTHER_HCA_CAP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												04h				
																												08h-0Ch				
other_hca_capabilities (See Table 1635, “Other HCA Capabilities Layout,” on page 1809)																												10h-4Ch				

Table 1634 - QUERY_OTHER_HCA_CAP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	other_hca_capabilities (See Table 1635, “Other HCA Capabilities Layout,” on page 1809)	Other HCA Capabilities.	

Table 1635 - Other HCA Capabilities Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
roce																																00h
																																04h-3Ch

Table 1636 - Other HCA Capabilities Field Descriptions

Offset	Bits	Name	Description	Access
00h	31	roce	RoCE field in HCA_CAP. Valid only when HCA_CAP.access_other_hca_roce==1	

29.3.7 MODIFY_OTHER_HCA_CAP - Modify Other HCA Capabilities

Table 1637 - MODIFY_OTHER_HCA_CAP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																function_id																08h
field_select																																0Ch
other_hca_capabilities (See Table 1635, “Other HCA Capabilities Layout,” on page 1809)																																10h-4Ch

Table 1638 - MODIFY_OTHER_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:6	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1638 - MODIFY_OTHER_HCA_CAP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	15:0	function_id	The function that the memory is given to the device for: 0: When accessing my HCA. 0xffff: Access all function_ids. others: function_id of specific vHCA used by vhca_group_manager (HCA_CAP.vhca_group_manager ==1).	
0Ch	31:0	field_select	Field select bitmask indicates which fields to modify in the other hca cap. Bit 0: RoCE	
10h- 4Ch	512	other_hca_capabilities (See Table 1635, “Other HCA Capabilities Layout,” on page 1809)	Other HCA Capabilities.	

Table 1639 - MODIFY_OTHER_HCA_CAP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																															04h	
																															08h-0Ch	

Table 1640 - MODIFY_OTHER_HCA_CAP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		

29.3.8 QUERY_SF_PARTITIONS - Query Sub-Function Partition Configuration

The QUERY_SF_PARTITIONS command returns the sub-function partition configuration, and is supported only when HCA_CAP.sf == 1.

For more details, please refer to Section 14.3, “Sub-Function,” on page 845.

Table 1641 - QUERY_SF_PARTITIONS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																														08h-0Ch		

Table 1642 - QUERY_SF_PARTITIONS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1643 - QUERY_SF_PARTITIONS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												04h				
																						num_sf_partitions						08h				
																												0Ch				
sf_partition[0](See Table 1645, “SF_PARTITION Output Structure Layout,” on page 1812)																												10h				
sf_partition[1]																												14h				
...																												18h-1Ch				

Table 1644 - QUERY_SF_PARTITIONS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	status		

Table 1644 - QUERY_SF_PARTITIONS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
08h	7:0	num_sf_partitions	Number of sub_vhca groups.	
10h-..	31:0	sf_partition[...](See Table 1645, “SF_PARTITION Output Structure Layout,” on page 1812)	sf_partitions list. The list size is indicated in num_sf_partitions	

Table 1645 - SF_PARTITION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																log_num_sf						log_sf_bar_size						00h				

Table 1646 - SF_PARTITION Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:8	log_num_sf	log(base 2) of num of sub-functions in this partition.	
	7:0	log_sf_bar_size	log(base 2) of bar size of each sub-function in this partition. In set, must be $\geq$ HCA_CAP.log_min_sf_size, and must be aligned in granularity of 4KB.	

29.3.9 SET_SF_PARTITIONS - Set Sub-Function Partition Configuration

The SET_SF_PARTITIONS command modifies the sub-function partition configuration and is supported only when HCA_CAP.sf_set_partition == 1.

For more details, please refer to Section 14.3, “Sub-Function,” on page 845.

Table 1647 - SET_SF_PARTITIONS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																								num_sf_partitions				08h				
																																0Ch
sf_partition[0](See Table 1645, “SF_PARTITION Output Structure Layout,” on page 1812)																																10h

Table 1647 - SET_SF_PARTITIONS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sf_partition[1]																																14h
...																																18h...

Table 1648 - SET_SF_PARTITIONS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	num_sf_partitions	Number of sub_vhca groups.	
10h-..	31:0	sf_partition[...](See Table 1645, “SF_PARTITION Output Structure Layout,” on page 1812)	sf_partitions list. The list size is indicated in num_sf_partitions	

Table 1649 - SET_SF_PARTITIONS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1650 - SET_SF_PARTITIONS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 1650 - SET_SF_PARTITIONS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		

29.3.10ALLOC_SF - Allocate Sub-Function

The ALLOC_SF command creates a sub-function on a specific sub-function partition, and is supported only when HCA_CAP.sf==1 and HCA_CAP.max_num_sf_partitions.

For more details, please refer to Section 14.3, “Sub-Function,” on page 845.

Table 1651 - ALLOC_SF Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																function_id																08h
																																0Ch

Table 1652 - ALLOC_SF Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	15:0	function_id	Sub_function ID. See Section 14.3, “Sub-Function,” on page 845	

Table 1653 - ALLOC_SF Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 1653 - ALLOC_SF Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 08h-0Ch

Table 1654 - ALLOC_SF Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		

29.3.11 DEALLOC_SF - De-Allocate Sub-Function

The DEALLOC_SF command destroys a sub-function, and it is supported only when HCA_CAP.sf == 1 and HCA_CAP.max_num_sf_partitions. **For more details, please refer to Section 14.3, “Sub-Function,” on page 845.**

Table 1655 - DEALLOC_SF Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																function_id																08h
																																0Ch

Table 1656 - DEALLOC_SF Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:1 6	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1656 - DEALLOC_SF Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	15:0	function_id	Sub_function ID. See Section 14.3, “Sub-Function,” on page 845.	

Table 1657 - DEALLOC_SF Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1658 - DEALLOC_SF Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		

29.3.12 QUERY_ADAPTER – Query Adapter

The QUERY_ADAPTER command retrieves adapter specific parameters. This information is used by the driver in order to clear interrupt signaling by the device (for more details, see *clr_int* in Table 14, “Initialization Segment,” on page 234). The data returned in the output mailbox is summarized in Table 1663, “QUERY_ADAPTER Parameters Block Layout” on page 1818.

Table 1659 - QUERY_ADAPTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1660 - QUERY_ADAPTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 1661 - QUERY_ADAPTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																04h				
																																08-0Ch				
query_adapter_struct (See Table 1663, “QUERY_ADAPTER Parameters Block Layout,” on page 1818)																																10h-10Ch				

Table 1662 - QUERY_ADAPTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-10Ch	2048	query_adapter_struct (See Table 1663, “QUERY_ADAPTER Parameters Block Layout,” on page 1818)		

Table 1663 - QUERY_ADAPTER Parameters Block Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
intapin																																10h
																																14h
								ieee_vendor_id																								18h
																vsd_vendor_id																1Ch
vsd																																20h-ECh
vsd_contd_psid																																F0h-FCh

Table 1664 - QUERY_ADAPTER Parameters Block Field Descriptions (Sheet 1 of 2)

Offset	Bits	Name	Description	Access
10h	31:24	intapin	[Internal] This field is deprecated When PCIe interrupt messages are being used, this value is used for clearing an interrupt. To clear an interrupt, the driver should write the value (1<<intapin) into the clr_int register. When using MSI-X, this register is not used.	
18h	23:0	ieee_vendor_id	IEEE vendor_id. Supported only starting from ISSI==1.	
1Ch	15:0	vsd_vendor_id	PCISIG Vendor ID (www.pcisig.com/membership/vid_search) of the vendor specifying/formatting the VSD. The vsd_vendor_id identifies the management domain of the vsd/psid data. Different vendors may choose different vsd/psid format and encoding as long as they use their assigned vsd_vendor_id. The psid format as described below is used in conjunction with Mellanox vsd_vendor_id (15B3h).	
20h-ECh	1664	vsd	Vendor Specific Data. The VSD string that is burnt to the Flash with the Firmware.	

Table 1664 - QUERY_ADAPTER Parameters Block Field Descriptions (Sheet 2 of 2)

Offset	Bits	Name	Description	Access
F0h-FCh	128	vsd_contd_psid	This field carries the last sixteen bytes of the VSD field. For Mellanox formatted VSD (vsd_vendor_id=15B3h) the last 16 bytes of VSD are used as PSID. The PSID field is a 16-ascii (byte) character string which acts as an HCA Adapter Card ID. The format of the PSID is as follows: Vendor Symbol (VS) - 3 characters Board Type Symbol (BT) - 3 characters Board Version Symbol (BV) - 3 characters Parameter Set Number (PS) - 4 characters Reserved (R) - 3 characters (filled with zeros) The various characters are organized as described in See Table 1665, “PSID Character Offsets (Example is in Parentheses),” on page 1819. Example: A PSID for Mellanox’s MHXL-CF128-T HCA board is MT_0030000001, where: MT_ is the Mellanox Vendor Symbol 003 is the MHXL-CF128-T Board Type Symbol 000 is the Board Version Symbol 0001 is the Parameter Set Number The byte order is as follows: Bits 31:24 at offset 0 represent the first character (byte), bits 23:16 - second character, etc. Bits 31:24 at offset 4 represent the fourth character, etc.	

Table 1665 - PSID Character Offsets (Example is in Parentheses)

000F0h	000F1h	000F2h	000F3h
VS1 (‘M’)	VS2 (‘T’)	VS3 (‘_’)	BT1 (‘3’)
000F4h	000F5h	000F6h	000F7h
BT2 (‘0’)	BT3 (‘0’)	BV1 (‘0’)	BV2 (‘0’)
000F8h	000F9h	000FAh	000FBh
BV3 (‘0’)	PS1 (‘1’)	PS2 (‘0’)	PS3 (‘0’)
000FCh	000FDh	000FEh	000FFh
PS4 (‘0’)	R1 (00000000b)	R2 (00000000b)	R3 (00000000b)

29.3.13 INIT_HCA – INIT HCA

This command initiates and opens the HCA.

Table 1666 - INIT_HCA Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																																08h
																parent_vhca_id																0Ch
sw_owner_id																																10h-1Ch

Table 1667 - INIT_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		

Table 1667 - INIT_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31	m_pci_port_child	If set, indicates this vHCA behave as a child in multi pci port. See Section 14.9, “Multi PCI Port,” on page 857	
	30	m_vhca_mk_child	If set, indicates this vHCA behave as a child in multi vHCA mkey. See Section 14.8, “Multi vHCA Mkey,” on page 856	
	15:0	parent_vhca_id	vHCA id of the parent in multi pci port / multi vHCA mkey. valid only when m_pci_port_child==1 or m_vhca_mk_child==1. The INIT_HCA.sw_owner_id of this vHCA and of the parent vHCA denoted by this field must be equal. See Section 14.8, “Multi vHCA Mkey,” on page 856 and Section 14.9, “Multi PCI Port,” on page 857 .	
10-1Ch	128	sw_owner_id	Software owner identifier. An identifier that set by software to inform the device that multiple VHCAs belong to same owner. Supported only when HCA_CAP.sw_owner_id==1.	

Table 1668 - INIT_HCA Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08-0Ch

Table 1669 - INIT_HCA Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.14 TEARDOWN_HCA – Tear-down HCA

The command has three flavors:

- Graceful Close – The command releases all HCA allocated resources. The command will stop further execution of descriptors. It is the responsibility of the software to close all QPs/SQs/RQs, CQs and EQs and to unmap all the EQs, prior to executing the TEARDOWN_HCA command.
- Force Close – This command is executed by the software when it detects an internal error. Upon execution of this command the HCA does the following:
 - Stops access to the Uplink bus as a bus master.
- Moves the network ports to the Link down state. Fast teardown – See Section 7.3.1, “Fast Teardown,” on page 245.

After executing the close HCA in Force flavor, DoorBells will still be consumed by the HCA, and they will be silently dropped.

Table 1670 - TEARDOWN_HCA Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																profile																08h
																																0Ch

Table 1671 - TEARDOWN_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	15:0	profile	0x0: Graceful_close 0x1: Force_close - Supported only when HCA_CAP.force_teardown==1 in best effort. SW needs to check the return state field returned by the command. Note that when doing force_close there will be no event on this command and SW must check the status in the command mailbox. 0x2: prepare_fast_teardown - Supported only when HCA_CAP.fast_teardown==1. SW needs to check the return state field returned by the command.	

Table 1671 - TEARDOWN_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	reserved		

Table 1672 - TEARDOWN_HCA Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																08h
																															state	0Ch

Table 1673 - TEARDOWN_HCA Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	0	state	Indicates the tear down with force_close state and the prepare_fast_tearardown state. 0x0: success 0x1: fail	

29.3.15ENABLE_HCA

This command must be used during driver start-up as described in Section 7.2, “HCA Driver Start-up,” on page 244.

Table 1674 - ENABLE_HCA Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h

Table 1674 - ENABLE_HCA Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
embedded cpu function																function_id																08h
																																0ch

Table 1675 - ENABLE_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	31	embedded_cpu_function	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on external Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU INIT_SEGMENT.embedded_cpu==1	
	15:0	function_id	function_id of the device to enable. 0: when accessing my function or when ECPF access PF function. others: when vhca_group_manager (HCA - CAP.vhca_group_manager ==1) accesses another function in its group.	

Table 1676 - ENABLE_HCA Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h

Table 1677 - ENABLE_HCA Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	reserved		

29.3.16DISABLE_HCA

This command must be used during driver teardown as described in Section 7.2, “HCA Driver Start-up,” on page 244.

Table 1678 - DISABLE_HCA Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
embedded cpu function																function_id																08h
																																0Ch

Table 1679 - DISABLE_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		

Table 1679 - DISABLE_HCA Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	embedded_cpu_function	Indicates if the function on Host or on the embedded CPU. 0x0: HOST - Function on external Host 0x1: EMBEDDED_CPU - Function on the embedded CPU. Supported only for functions on the embedded CPU INIT_SEGMENT.embedded_cpu==1	
	15:0	function_id	function_id of the device to enable. 0: when accessing my function or when ECPF access PF function. others: when vhca_group_manager (HCA - CAP.vhca_group_manager ==1) accesses another function in its group.	

Table 1680 - DISABLE_HCA Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h

Table 1681 - DISABLE_HCA Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	reserved		

29.3.17QUERY_ISSI

This command is used to query the supported Interface Step Sequence ID. If this command returns a “BAD_OPCODE”, it indicates that only ISSI = 0 is supported.

SW must use this command to query the SUPPORTED_ISSI, and then use the SET_ISSI command to set the actual ISSI it prefers to run with, which should be the minimum between its ISSI

and the maximum supported_issi of the device. See Section 8.1, “ISSI - Interface Step Sequence ID,” on page 248.

Table 1682 - QUERY_ISSI Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1683 - QUERY_ISSI Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1684 - QUERY_ISSI Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																04h				
																current_issi																08h				
																																0Ch-1C				
supported_issi																																20h-6Ch				

Table 1685 - QUERY_ISSI Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	15:0	current_issi	Current Interface Step Sequence ID. Can be changed using SET_ISSI command.	

Table 1685 - QUERY_ISSI Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
20h-6Ch	640	supported_issi	A bitmask field indicates which ISSI is supported by the device: Bit 0: indicates if ISSI =0 supported. Bit 1: indicates if ISSI =1 supported. Bit 2: indicates if ISSI =2 supported. Bit N: indicates if ISSI =N supported.	

29.3.18SET_ISSI

Table 1686 - SET_ISSI Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																current_issi																08h
																																0Ch

Table 1687 - SET_ISSI Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	15:0	current_issi	Current Interface Step Sequence ID to work with. It is recommended to set it to min (my issi, max device supported_issi)	

Table 1688 - SET_ISSI Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1689 - SET_ISSI Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.3.19SET_DRIVER_VERSION

This command is used to set the version of driver which performs start-up over the device.

When HCA_CAP.driver_version==1, the driver which performs start-up should set its version using this command in this stage as indicated in Section 7.2, “HCA Driver Start-up,” on page 244.

Table 1690 - SET_DRIVER_VERSION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0C
driver_version																																10h-4C

Table 1691 - SET_DRIVER_VERSION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1691 - SET_DRIVER_VERSION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-4Ch	512	driver_version	The Driver version string is a concatenation of 3 fields separated by “,”: - OS_name - generic string - Driver_name - Generic string - Driver_version_number - Concatenation of 3 numbers separated by “.”: - Main_version - 3 digits (no need for leading zeros) - Minor_version - 3 digits (mandatory leading zeros) - Sub_minor_version - 6 digits (mandatory leading zeros) The order of the fields is from left (first - OS_name) to right (last - Sub_minor_version) The total length of the driver_version is limited to 64 Bytes (including the end-of-string symbol).	

Table 1692 - SET_DRIVER_VERSION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																04h
																																08h-0Ch

Table 1693 - SET_DRIVER_VERSION Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.3.20QUERY_DRIVER_VERSION

This command is used to read the driver version that was set by SET_DRIVER_VERSION. The command is supported if QUERY_HCA_CAP.query_driver_version ==1. The command can also be used by the vhca_group_manager to query the driver versions of its associated group members.

Table 1694 - QUERY_DRIVER_VERSION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_function																function_id																08h

Table 1695 - QUERY_DRIVER_VERSION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_function	If set, indicates accessing capabilities of other function identified by function_id. supported only for vhca resource manager (HCA_CAP.vhca_resource_manager==1)	
	15:0	function_id	function_id of vHCA: 0x0: When accessing my HCA. Others: Function_id of specific vHCA- used-by vhca_group_manager (HCA_CAP.vhca_group_manager==1).	

Table 1696 - QUERY_DRIVER_VERSION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1696 - QUERY_DRIVER_VERSION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
driver_version																																10h-4Ch

Table 1697 - QUERY_DRIVER_VERSION Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	driver_version	See description in SET_DRIVER_VERSION command.	

29.3.21ALLOC_PD - Allocate Protection Domain

Table 1698 - ALLOC_PD Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1699 - ALLOC_PD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 1700 - ALLOC_PD Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								pd																								08h
																																0Ch

Table 1701 - ALLOC_PD Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	pd	Protection Domain	
0Ch	31:0	reserved		

29.3.22 DEALLOC_PD - De-Allocate Protection Domain

Table 1702 - DEALLOC_PD Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									pd																			08h				
																																0Ch

Table 1703 - DEALLOC_PD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	pd	Protection Domain	
0Ch	31:0	reserved		

Table 1704 - DEALLOC_PD Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1705 - DEALLOC_PD Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.23 ALLOC_UAR - Allocate UAR

Table 1706 - ALLOC_UAR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1707 - ALLOC_UAR Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod		

Table 1708 - ALLOC_UAR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
								uar																								08h
																																0Ch

Table 1709 - ALLOC_UAR Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	uar	UAR which is allocated. In case HCA_CAP.uar_4kb==1, then this is the first 4kB UAR allocated of HCA_CAP.number_of_uars_per_page out of total UARs.	

29.3.24 DEALLOC_UAR - De-Allocate UAR

Table 1710 - DEALLOC_UAR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								uar																								08h
																																0Ch

Table 1711 - DEALLOC_UAR Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod		
08h	23:0	uar	UAR	
0Ch	31:0	reserved		

Table 1712 - DEALLOC_UAR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1713 - DEALLOC_UAR Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.25ALLOC_XRCD - Allocate XRC Domain

Table 1714 - ALLOC_XRCD Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1715 - ALLOC_XRCD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 1716 - ALLOC_XRCD Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								xrcd																								08h
																																0Ch

Table 1717 - ALLOC_XRCD Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	xrcd	XRC domain	
0Ch	31:0	reserved		

29.3.26 DEALLOC_XRCD - De-allocate XRC Domain

Table 1718 - DEALLOC_XRCD Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									xrcd																		08h					
																																0Ch

Table 1719 - DEALLOC_XRCD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrcd	XRC domain	

Table 1719 - DEALLOC_XRCD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	reserved		

Table 1720 - DEALLOC_XRCD Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				004h
																																				08h-0Ch

Table 1721 - DEALLOC_XRCD Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.27 CONFIG_INT_MODERATION - Configure Interrupt Moderation

The command sets a maximal interrupt frequency. When MSI-X is enabled, the setting is per MSI-X vector. When MSI-X is disabled, the setting is for the PCIe interrupt pin emulation (INT message).

Table 1722 - CONFIG_INT_MODERATION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
				min_delay												int_vector																08h
																																0Ch

Table 1723 - CONFIG_INT_MODERATION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: write 1: read	
08h	27:16	min_delay	Minimal interval (in multiples of 1usec) between interrupts	
	15:0	int_vector	If MSI-X is enabled, this field holds the MSI-X vector. Otherwise, this field is reserved.	
0Ch	31:0	reserved		

Table 1724 - CONFIG_INT_MODERATION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
				min_delay												int_vector																08h
																																0Ch

Table 1725 - CONFIG_INT_MODERATION Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	27:16	min_delay	Minimal interval (in multiples of 1usec) between interrupts	
	15:0	int_vector	If MSI-X is enabled, this field holds the MSI-X vector. Otherwise, this field is reserved.	
0Ch	31:0	reserved		

29.3.28 SET_BURST_SIZE

The SET_BURST_SIZE command is used to configure device Burst_Size which defines the number of bytes that are sent before rescheduling, i.e start working on next schedule queue.

Table 1726 - SET_BURST_SIZE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																device_burst_size																0Ch

Table 1727 - SET_BURST_SIZE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	22:0	device_burst_size	Number of bytes sent before rescheduling (i.e going to next schedule queue).	

Table 1728 - SET_BURST_SIZE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
																syndrome																004h
																																08h-0Ch

Table 1729 - SET_BURST_SIZE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.3.29 QUERY_BURST_SIZE

The SET_BURST_SIZE command is used to query device Burst_Size which defines the number of bytes that will be sent before rescheduling, i.e start working on next schedule queue.

Table 1730 - QUERY_BURST_SIZE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1731 - QUERY_BURST_SIZE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1732 - QUERY_BURST_SIZE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																00h																								
syndrome																																004h																								
																																08h-0Ch																								
								device_burst_size																								0Ch																								

Table 1733 - QUERY_BURST_SIZE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	22:0	device_burst_size	Number of bytes will be sent before rescheduling (i.e going to next schedule queue).	

29.3.30 ACTIVATE_TRACER – Activate the tracer

ACTIVATE_TRACER command used to enable the tracer. When enabling the tracer, software should inform the hardware about the mkey where to dump the trace.

It is the software's responsibility to disable the tracer using DEACTIVATE_TRACER command after finishing using it.

Note that the tracer can be used only by one client at the same time, so if ACTIVATE_TRACER command will be called while the tracer is active, the command will fail and return BAD_RES_STATE.

Table 1734 - ACTIVATE_TRACER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
mkey																																08h
																																0Ch

Table 1735 - ACTIVATE_TRACER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	mkey	Memory key	
0Ch	31:0	reserved		

Table 1736 - ACTIVATE_TRACER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1737 - ACTIVATE_TRACER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.31 DEACTIVATE_TRACER – Deactivate the tracer

Table 1738 - DEACTIVATE_TRACER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
mkey																																08h
																																0Ch

Table 1739 - DEACTIVATE_TRACER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	mkey	Memory key	
0Ch	31:0	reserved		

Table 1740 - DEACTIVATE_TRACER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h
syndrome																																	004h
																																	08h-0Ch

Table 1741 - DEACTIVATE_TRACER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.32ALLOC_TRANSPORT_DOMAIN - Allocate Transport Domain

Table 1742 - ALLOC_TRANSPORT_DOMAIN Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1743 - ALLOC_TRANSPORT_DOMAIN Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 1744 - ALLOC_TRANSPORT_DOMAIN Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								transport_domain																								08h
																																0Ch

Table 1745 - ALLOC_TRANSPORT_DOMAIN Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	transport_domain	Transport Domain ID	
0Ch	31:0	reserved		

29.3.33 DEALLOC_TRANSPORT_DOMAIN - De-Allocate Transport Domain

Table 1746 - DEALLOC_TRANSPORT_DOMAIN Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								transport_domain																								08h
																																0Ch

Table 1747 - DEALLOC_TRANSPORT_DOMAIN Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	transport_domain	Transport Domain ID.	
0Ch	31:0	reserved		

Table 1748 - DEALLOC_TRANSPORT_DOMAIN Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 1748 - DEALLOC_TRANSPORT_DOMAIN Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 1749 - DEALLOC_TRANSPORT_DOMAIN Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.34SET_WOL_ROL – Set WOL and ROL Mode

SET_WOL_ROL is used to configure the mode of Wake on LAN and Reset on LAN. See Section 7.5, on page 246 and Section 7.6, on page 246.

The default setting of WOL/ROL mode can be configured using non-volatile setting. The setting, which is configured through the following command, is volatile and not retained through power cycle to ConnectX device.

Table 1750 - SET_WOL_ROL Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
rol mode valid		wol mode valid														rol_mode								wol_mode								08h
																																0Ch

Table 1751 - SET_WOL_ROL Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 1751 - SET_WOL_ROL Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	31	rol_mode_valid	Must be set to 1 when modifying rol_mode.	
	30	wol_mode_valid	Must be set to 1 when modifying wol_mode.	
	15:8	rol_mode	Bitmask field indicates which Reset on LAN modes to be enabled/disabled (valid when rol_mode_valid == 1). Bit 0: reset_on_magic_packet - Supported only when HCA_CAP.rol_g==1. Bit 1: reset_on_secured_magic_packet - Supported only when HCA_CAP.rol_s==1.	
	7:0	wol_mode	Bitmask field indicates which Wake on LAN modes to be enabled/disabled (valid when wol_mode_valid == 1). Bit 1: wake_on_secured_magic_packet - Supported only when HCA_CAP.wol_s==1. Bit 2: wake_on_magic_packet - Supported only when HCA_CAP.wol_g==1. Bit 3: wake_on_arp - Supported only when HCA_CAP.wol_a==1. Bit 4: wake_on_broadcast_packet - Supported only when HCA_CAP.wol_b==1. Bit 5: wake_on_multicast_packet - Supported only when HCA_CAP.wol_m==1. Bit 6: wake_on_unicast_packet - Supported only when HCA_CAP.wol_u==1. Bit 7: wake_on_phy_activity - Supported only when HCA_CAP.wol_p==1.	

Table 1752 - SET_WOL_ROL Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1753 - SET_WOL_ROL Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 1753 - SET_WOL_ROL Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.3.35 QUERY_WOL_ROL – Query WOL and ROL Mode

QUERY_WOL_ROL used to query the mode of Wake on LAN and Reset on LAN. See Section 7.5, on page 246 and Section 7.6, on page 246.

Table 1754 - QUERY_WOL_ROL Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																														08h-0Ch		

Table 1755 - QUERY_WOL_ROL Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1756 - QUERY_WOL_ROL Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																														004h		
																rol_mode								wol_mode								08h
																														0Ch		

Table 1757 - QUERY_WOL_ROL Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	15:8	rol_mode	Bitmask field indicates which Reset on LAN modes to be enabled/disabled (valid when rol_mode_valid ==1). Bit 0: reset_on_magic_packet - Supported only when HCA_CAP.rol_g==1. Bit 1: reset_on_secured_magic_packet - Supported only when HCA_CAP.rol_s==1.	
	7:0	wol_mode	Bitmask field indicates which Wake on LAN modes to be enabled/disabled Bit 1: wake_on_secured_magic_packet - Supported only when HCA_CAP.wol_s==1. Bit 2: wake_on_magic_packet - Supported only when HCA_CAP.wol_g==1. Bit 3: wake_on_ARP - Supported only when HCA_CAP.wol_a==1. Bit 4: wake_on_broadcast_packet - Supported only when HCA_CAP.wol_b==1. Bit 5: wake_on_multicast_packet - Supported only when HCA_CAP.wol_m==1. Bit 6: wake_on_unicast_packet - Supported only when HCA_CAP.wol_u==1. Bit 7: wake_on_phy_activity - Supported only when HCA_CAP.wol_p==1.	

29.4 LAG Commands

The commands below enable to create and manage LAG. See [Chapter 14, “Virtualization”](#) for further details.

Note that these commands must be performed when all VFs are down.

29.4.1 CREATE_LAG

Table 1758 - CREATE_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h

Table 1758 - CREATE_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)																															08h-0Ch	

Table 1759 - CREATE_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	
08h-0Ch	64	lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)	LAG Context.

Table 1760 - CREATE_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1761 - CREATE_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

Table 1762 - LAG CONTEXT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fdb_selection_mode																												lag_state		00h		
																				tx_remap_affinity_2						tx_remap_affinity_1		04h				

Table 1763 - LAG CONTEXT Field Descriptions

Offset	Bits	Name	Description
00h	31	fdb_selection_mode	0x0: BY_AFFINITY - According to QP/TIS.lag_tx_port_affinity, subject to tx_remap_affinity. If the QP/TIS is affined (by to QP/TIS.lag_tx_port_affinity, subject to tx_remap_affinity) port1, then go to FDB of port 1, else go to FDB of port 2. 0x1: NATIVE_FDB - Go to the FDB natively associated with the vhca. Supported only when HCA_CAP.lag_native_fdb_selection==1.
	2:0	lag_state	Indicates LAG state 0x0: disable 0x1: enabled Valid only in QUERY_LAG command.
04h	11:8	tx_remap_affinity_2	Indicates the remapping port number of lag_tx_port_affinity == 2.
	3:0	tx_remap_affinity_1	Indicates the remapping port number of lag_tx_port_affinity == 1.

29.4.2 MODIFY_LAG

Table 1764 - MODIFY_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
field_select																																0Ch

Table 1764 - MODIFY_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)																																10h-14h

Table 1765 - MODIFY_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	
0Ch	31:0	field_select	Field Select indicates which fields to modify Bit 0: tx_port_remap
10h-14h	64	lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)	LAG Context.

Table 1766 - MODIFY_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																00h	
syndrome																																	04h
																																	08h-0Ch

Table 1767 - MODIFY_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.4.3 QUERY_LAG

Table 1768 - QUERY_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1769 - QUERY_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	

Table 1770 - QUERY_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)																																08h-0Ch

Table 1771 - QUERY_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	lag_context (See Table 1762, “LAG CONTEXT Structure Layout,” on page 1851)	LAG Context.

29.4.4 DESTROY_LAG

Table 1772 - DESTROY_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1773 - DESTROY_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	

Table 1774 - DESTROY_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1775 - DESTROY_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.4.5 CREATE_VPORT_LAG

CREATE VPORT LAG command merges the flow steering lookup of all LAG Vports into the master’s flow steering lookup. This function is allowed only by the LAG master. For details, see Chapter 14, “Virtualization”.

Table 1776 - CREATE_VPORT_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1777 - CREATE_VPORT_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	

Table 1778 - CREATE_VPORT_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1779 - CREATE_VPORT_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.4.6 DESTROY VPORT LAG

Table 1780 - DESTROY_VPORT_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1781 - DESTROY_VPORT_LAG Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	

Table 1782 - DESTROY_VPORT_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																00h																								
syndrome																																04h																								
																																08h-0Ch																								

Table 1783 - DESTROY_VPORT_LAG Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.5 Registers Access Commands

29.5.1 ACCESS_REGISTER

The ACCESS_REGISTER command is used by the driver to access registers for chip configuration, etc.

Table 1784 - ACCESS_REGISTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																register_id																08h
argument																																0Ch
register_data[0]-																																10h
register_data[1]-																																14h
...																																...

Table 1785 - ACCESS_REGISTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Opcode modifier 0: WRITE 1: READ	
08h	15:0	register_id	ID of the register being accessed	
0Ch	31:0	argument	The meaning of this field is defined per register_id	
10h-...	31:0	register_data[...]	For write - register data For read - reserved	

Table 1786 - ACCESS_REGISTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 1786 - ACCESS_REGISTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08-0Ch
register_data[0]																																10h
register_data[1]																																14h
...																																...

Table 1787 - ACCESS_REGISTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	31:0	reserved		
10h-...	31:0	register_data[...]	For write - reserved For read - register data	

[FG\]. Should have inline option](#)

29.5.2 ACCESS_REG_SPACE

The ACCESS_REG_SPACE command is used by the driver to access registers for chip configuration etc.

The device register space is described in Chapter 28, “Command Registers” on page 1080.

Table 1788 - ACCESS_REG_SPACE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																																00h
																op_mod																04h
address																																08h
																length																0Ch
register data																																10h...

Table 1789 - ACCESS_REG_SPACE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod	Opcode modifier: 0 - write 1 - read	
08h	31:0	address	Address of register to be accessed	
0Ch	15:0	length	Length of data in dwords Legal values depend on register type	
10h-...		register data	For write - register data For read - reserved	

Table 1790 - ACCESS_REG_SPACE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
register data																																10h...

Table 1791 - ACCESS_REG_SPACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		
10h-...		register data	For write - reserved For read - register data	

29.5.3 ACCESS_REG_SPACE_DWORD

The ACCESS_REG_SPACE_DWORD command is used by the driver to access registers for chip configuration etc. in DWORD granularity

The device register space is described in Chapter 28, “Command Registers” on page 1080.

Table 1792 - ACCESS_REG_SPACE_DWORD Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																												00h				
																op_mod												04h				
address																																08h
register data																																0Ch

Table 1793 - ACCESS_REG_SPACE_DWORD Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod	Opcode modifier: 0 - write 1 - read	
08h	31:0	address	address of register to be accessed	
0Ch	31:0	register data	For write - register data For read - reserved	

Table 1794 - ACCESS_REG_SPACE_DWORD Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
register data																																0Ch

Table 1795 - ACCESS_REG_SPACE_DWORD Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	reserved		
0Ch	31:0	register data	For write - reserved For read - register data	

29.6 TPT Commands

TPT commands initialize and configure address translation and protection structures required for the HCA.

To register a Memory Region, software should allocate and configure MKey context by posting ALLOC_MKEY command, passing MKey parameters and in return receiving MKey handle (number) assigned by HW. To de-register a Memory Region, software should use the DEALLOC_MKEY command on the appropriate MKey handle. It is recommended that, prior to releasing a Memory Region, software will deallocate indirect memory regions pointing to it.

The following are the commands affecting the Translation and Protection (TPT) mechanism and its Memory Protection Table (MKey).

Table 1796 - TPT Commands

Mnemonic	Command	Description	Reference
CREATE_MKEY	Create MKey entry	Generates new MKey	Section 29.6.1, on page 1861
QUERY_MKEY	Query MKey entry	Gets a snapshot of MKey context.	Section 29.6.2, on page 1865
DESTROY_MKEY	Destroy MKey entry	Destroy MKey. After this command has been executed, accesses to this key will result in an access fault.	Section 29.6.3, on page 1868
QUERY_SPECIAL_CONTEXTS	Query special context numbers	Returns a context number to be used for special purposes	Section 29.6.5, on page 1871

29.6.1 CREATE_MKEY – Create MKey Entry

The CREATE_MKEY command generates new MKeys. The context of the new MKey object is taken from the input mailbox. HW provides an MKey index as an output parameter. This index will be used by SW as a handle when accessing this object.

Table 1797 - CREATE_MKEY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								input_mkey_index																								08h
mkey_unmem_valid																														0Ch		
pg_access																																

Table 1797 - CREATE_MKEY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
memory_key_mkey_entry (See Table 362, “MKey Context Format,” on page 629)																																10h-4Ch
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
e_bsf_pointer (See Table 1805, “e-bsf_pointer Layout,” on page 1865)																																58h-5Ch
translations_octword_actual_size																																60h
mkey_umem_id																																64h
mkey_umem_offset																																68h-6Ch
																bsf_octword_actual_size																70h
																																74h-10Ch
klm_or_pas_or_mtt_bsf[0]																																110h
klm_or_pas_or_mtt_bsf[1]																																114h
...																																...

Table 1798 - CREATE_MKEY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	input_mkey_index	mkey index of the created mkey. 0 means reserved and mkey index is returned by the device in command output. Supported only when HCA_CAP.mkey_by_name==1. See Table 1801, “MKey Most Significant Bits Selection,” on page 1864.	

Table 1798 - CREATE_MKEY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31	pg_access	Per-page access rights. If set, the <i>wr_en</i> and <i>rd_en</i> fields of provided translation entries are valid and must specify the desired access rights. Can be set only when access_mode==MTT.	
	30	mkey_umem_valid	Indicates whether the buffer is given directly by MTT list or by umem_id and umem_offset. When 0 - buffer given by MTT/KLM list When 1 - buffer given by umem_offset and umem_offset	
10h-4Ch	512	memory_key_mkey_entry (See Table 362, “MKey Context Format,” on page 629)	MKey context.	
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. Used instead of <i>pas</i> when global <i>e</i> bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803). mtt index should be 64 byte aligned (mtt_index_l[2:0] should be 0x0).	
58h-5Ch	64	e_bsf_pointer (See Table 1805, “e-bsf_pointer Layout,” on page 1865)		
60h	31:0	translations_octword_actual_size	Actual number of octwords that contain translation entries. Can be 0 if no KLMs/MTTs are delivered.	
64h	31:0	mkey_umem_id	UMEM ID of the mkey. valid only when ~~uid!=0~~ mkey_umem_valid==1.	
68h-6Ch	64	mkey_umem_offset	~~In case uid!=0~~ Indicates offset in allocated umem id. valid only when mkey_umem_valid==1.	
70h	15:0	bsf_octword_actual_size	Actual number of octwords that contain BSF. BSF list should be aligned to 64B. Can be 0 if no BSFs are delivered. List of BSFs places after KLMs/PASs/MTTs list. Valid only when HCA_CAP.bsf_in_create_mkey==1.	
110h-...	31:0	klm_or_pas_or_mtt_bsf[...]	Translation entries and BSFs.	

Table 1799 - CREATE_MKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																											00h					

Table 1799 - CREATE_MKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																004h
								mkey_index																								08h
																																0Ch

Table 1800 - CREATE_MKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	mkey_index	MKey index	
0Ch	31:0	reserved		

When choosing MKey number, the two most significant bits should be chosen in the following way, according to desired Mkey properties. An Mkey can be configured to enable signature transactions. In addition, chip supports Transaction Status (or TRAST) to optimize transactions over large number of BSFs or KLMs. The TRAST remembers the current position within KLMs or BSFs and prevents the need to traverse all previous KLMs/BSFs to find the current position. Combinations missing from table below are not supported.

Table 1801 - MKey Most Significant Bits Selection

Signature	BSF TRAST	KLM TRAST	Mkey_index[23:22]
No	No	No	00
Yes	No	No	01
Yes	Yes	No	10
Yes	Yes	Yes	11

Table 1802 - e-mtt_pointer Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
reserved																								mtt_index_h				00h				
mtt_index_l																										04h						

Table 1803 - e-mtt_pointer Field Descriptions

Offset	Bits	Name	Description	Access
00h	6:0	mtt_index_h	MTT ICM address is $MTT\_base + 8 * mtt\_index$. For MKey to be accessed by UMR - should be 64B aligned. mtt_index_h[6:5] should follow the below rules (See Table 1804, “MTT Most Significant Bits Selection,” on page 1865)	
04h	31:3	mtt_index_l		

When choosing mtt_index, the most significant bits should be chosen in the following way:

Table 1804 - MTT Most Significant Bits Selection

MTT Type	mtt_index_h	mtt_index_l [28:27]	mtt_index_l [26:24]	
Data MTT	0x00	0x0	no restrictions	input for CREATE_MKEY with access_mode==MTT
KLM	0x40	0x0	no restrictions	input for CREATE_MKEY with access_mode==KLM
Control MTT	0x60	0x0	0x0	input for CREATE_MTT

Table 1805 - e-bsf_pointer Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																								bsf_index_h				00h				
bsf_index_l																												04h				

Table 1806 - e-bsf_pointer Field Descriptions

Offset	Bits	Name	Description	Access
00h	6:0	bsf_index_h	BSF ICM address is $BSF\_base + 16 * bsf\_index$. For MKey to be accessed by UMR - should be 64B aligned.	
04h	31:4	bsf_index_l		

29.6.2 QUERY_MKEY – Query MKey Entry

The QUERY_MKEY command requests a snapshot of an MKEY entry. The command takes the current state of an MKEY entry and stores it in the output mailbox. The command will fail if the requested MKey entry is invalid. The command also stores in the output mailbox the KLMs/PAs

or the BSFs as selected using the opcode modifier field below. The command will fill the pages provided by the software up to the mailbox size.

Table 1807 - QUERY_MKEY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								mkey_index																								08h
pg_access																															0Ch	

Table 1808 - QUERY_MKEY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0x0: PAS_KLMs - QUERY returns PAS/KLMs 0x2: BSFs - QUERY returns BSFs	
08h	23:0	mkey_index	MKey index	
0Ch	31	pg_access	Per-page access rights. If set, the <i>wr_en</i> and <i>rd_en</i> fields of provided translation entries are valid and return the current rights. Can be set only if Mkey created with pg_access ==1.	

Table 1809 - QUERY_MKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h
																																0Ch
memory_key_mkey_entry (See Table 362, “MKey Context Format,” on page 629)																																10h-4Ch

Table 1809 - QUERY_MKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																															50h-54h	
e_bsf_pointer (See Table 1805, “e-bsf_pointer Layout,” on page 1865)																															58h-5Ch	
																															60h-10Ch	
bsf0_or_klm0_or_pas_or_mtt0_1																															110h-11Ch	
bsf1_or_klm1_or_pas_or_mtt2_3																															120h-12Ch	

Table 1810 - QUERY_MKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	memory_key_mkey_entry (See Table 362, “MKey Context Format,” on page 629)	MKey context.	
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803).	
58h-5Ch	64	e_bsf_pointer (See Table 1805, “e-bsf_pointer Layout,” on page 1865)	Pointer to the BSFs in the ICM.	

Table 1810 - QUERY_MKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
110h-11Ch	128	bsf0/ _or_klm0_or_pas_or_mt t0_1	Translation entries and BSFs. When translation entries are not KLMs: - If the pg_access bit is set, translation entries are MTTs - If the pg_access bit not set, translation entries are PASs	
120h-12Ch	128	bsf1_or_klm1_or_pas_o r_mtt2_3	Translation entries and BSFs.	

29.6.3 DESTROY_MKEY – Destroy MKey Entry

Table 1811 - DESTROY_MKEY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								mkey_index																								08h
emd_on_behalf function_idvhea_id																																0Ch

Table 1812 - DESTROY_MKEY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	mkey_index	Memory Key Index	

Table 1813 - DESTROY_MKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h
syndrome																																004h	
																																08h-0Ch	

Table 1814 - DESTROY_MKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

The DESTROY_MKEY command invalidates the MKey entry.

29.6.4 PAGE_FAULT_RESUME

After generating a page fault event, the QP state changes so that it does not generate further page faults of the same type until the page fault is resolved by the software. Resolving page faults is done using the command PAGE_FAULT_RESUME, which also moves the QP to an error state if needed. Note that in some cases a page fault might have been reported speculatively. In such cases, an error resolution does not result in the QP moving to an error state.

Table 1815 - PAGE_FAULT_RESUME Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
error					page _ fault _ type		wq_number																							08h		
								token																							0Ch	

Table 1816 - PAGE_FAULT_RESUME Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 1816 - PAGE_FAULT_RESUME Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	error	Whether the resolution resolved in error (the page could not be mapped)	
	26:24	page_fault_type	Page fault type. 0x0: RMP related page fault 0x1: Requester QP WQE pagefault on send or write opcodes 0x2: Responder QP receive WQE pagefault, no SRQ involved 0x3: Requester QP WQE pagefault on read or atomic opcodes 0x4: RDMA Read responder pagefault (not valid). 0x5: Tag matching related pagefault 0x6: RDMA write responder pagefault (not valid). 0x7: SQ related page fault	
	23:0	wq_number	The work queue number for which the fault is being resolved, if available from the original event report. If not available, token should be reported here as well.	
0Ch	23:0	token	The token that was given in the page fault event must be set here. [Internal for FW]: This field might be zero with old drivers, in which case the value from wq_number should be used instead.	

Table 1817 - PAGE_FAULT_RESUME Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1818 - PAGE_FAULT_RESUME Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 1818 - PAGE_FAULT_RESUME Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h-0Ch	64	reserved		

29.6.5 QUERY_SPECIAL_CONTEXTS – Query Special Context Numbers

The QUERY_SPECIAL_CONTEXTS returns a context number to be used for special purposes such as, the DUMP_FILL Mkey and the Reserved Lkey.

Table 1819 - QUERY_SPECIAL_CONTEXTS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																																08h-0Ch

Table 1820 - QUERY_SPECIAL_CONTEXTS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h-0Ch	64	reserved		

Table 1821 - QUERY_SPECIAL_CONTEXTS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
dump_fill_mkey																																08h
resd_lkey																																0Ch

Table 1821 - QUERY_SPECIAL_CONTEXTS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
null_mkey																															10h	
terminate_scatter_list_mkey																															14h	
repeated_mkey																															18h	
																															1Ch	

Table 1822 - QUERY_SPECIAL_CONTEXTS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	dump_fill_mkey	Dump/fill mkey	
0Ch	31:0	resd_lkey	The value of the reserved Lkey for Base Memory Management Extension	
10h	31:0	null_mkey	The value of NULL_MKEY that is used to indicate non-present KLM/KSM entries, where it causes the device to generate page fault event when trying to access it. KLMs that reference NULL_MKEY should always use address 0. Supported only when HCA_CAP.null_mkey==1. See Table 1571, “HCA Capabilities Layout,” on page 1671.	
14h	31:0	terminate_scatter_list_mkey	Indicates a terminate scatter entry in a receive wqe scatter list entry. See Section 8.9.1.2, “Receive Queue,” on page 302 Supported only when HCA_CAP.terminate_scatter_list_mkey==1. Note that in case HCA_CAP.terminate_scatter_list_mkey==0, then SW must assume that the terminate_scatter_list_mkey value is 0x00000100. [Internal] FW must report by default 0x100 to keep backward compatibility	
18h	31:0	repeated_mkey	Identifies a repeated block format (See Table 706, “Repeated Block Format,” on page 919). Supported only when HCA_CAP.repeated_block_mkey==1. Note that in case HCA_CAP.repeated_block_mkey==0, then SW must assume that the repeated_block_mkey value is 0x400. [Internal] FW must report by default 0x400 to keep backward compatibility	

29.6.6 ALLOC_MEMIC – Allocate MEMIC

The ALLOC_MEMIC allocates a MEMIC buffer in a specific size. See Section 18.4, “MEMIC - Memory Mapped to InterConnect,” on page 923.

Table 1823 - ALLOC_MEMIC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																								log_memic_addr_align- ment								0Ch
range_start_addr																																10h-14h
range_size																																18h
memic_size																																1Ch

Table 1824 - ALLOC_MEMIC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	7:0	log_memic_addr_align- ment	The alignment of the allocated buffer is $64 \times 2^{\text{log\_memic\_addr\_alignment}}$ Bytes. Must be $\leq \text{HCA\_CAP.log\_max\_memic\_addr\_alignment}$	
10h-14h	64	range_start_addr	Start address of the range where to allocate the MEMIC buffer. Must be aligned to 64Bytes and $\geq \text{HCA\_CAP.DEIVDE\_MEMORY.memic\_bar\_start\_addr}$. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.	

Table 1824 - ALLOC_MEMIC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:0	range_size	Range size starting from range_start_addr where the MEMIC buffer should be allocated. Must be aligned to 64Bytes. Range_start_addr+range_size must be <= HCA_CAP.DEVICE_MEMORY.log_memic_bar_size. See Table 1605, “Device Memory Capabilities Layout,” on page 1785.	
1Ch	31:0	memic_size	Size of the MEMIC buffer to be allocated. Must be aligned to 64Bytes.	

Table 1825 - ALLOC_MEMIC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
memic_start_addr																																08h-0Ch

Table 1826 - ALLOC_MEMIC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	memic_start_addr	Start address on the bar where MEMIC buffer is allocated.	

29.6.7 DEALLOC_MEMIC – De-Allocate MEMIC

The DEALLOC_MEMIC de-allocates a MEMIC buffer. See Section 18.4, “MEMIC - Memory Mapped to InterConnect,” on page 923.

Note that the command will fail in case the deallocated range not fully allocated. i.e. in case part of the range is already free.

Table 1827 - DEALLOC_MEMIC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0C <h></h>
memic_start_addr																																10h-14h
memic_size																																18h
																																1Ch

Table 1828 - DEALLOC_MEMIC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-14h	64	memic_start_addr	Start address of the MEMIC buffer to be deallocated. Must be aligned to 64Bytes.	
18h	31:0	memic_size	Size of the MEMIC buffer to be deallocated. Must be aligned to 64Bytes.	

Table 1829 - DEALLOC_MEMIC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1830 - DEALLOC_MEMIC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.7 Signature Handover Commands

The following are the signature handover commands.

Table 1831 - Initialization and General Commands

Mnemonic	Command	Description	Reference
CREATE_PSV	Create PSV	Creates 1 or 4 PSVs and returns their PSV numbers to SW	Section 29.7.1, on page 1876
DESTROY_PSV	Destroy PSV	Destroys the PSV whose number is in the input parameter field	Section 29.7.2, on page 1878
QUERY_PSV	Query PSV	Returns the PSV context	Section 29.7.3, on page 1879

29.7.1 CREATE_PSV

CREATE_PSV creates 1 to 4 PSVs and returns their PSV numbers to SW. The layout of the output mailbox is shown in Table 1834, “CREATE_PSV Output Structure Layout”. When creating PSVs, it is recommended to create them in one command. For example, if 2 PSVs are needed, execute a single CREATE_PSV command with num_psv=2.

Command returns the PSV indices created. If 1 PSV was requested, only PSV0 is valid. If 2 PSVs were requested, PSV0 and PSV1 are valid and so on.

Table 1832 - CREATE_PSV Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
num_psv				end on behalf				pd																08h								
								psv0																0Ch								

Table 1833 - CREATE_PSV Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:28	num_psv	Number of PSVs requested (valid values are 1,2,3,4)	
	23:0	pd	Protection Domain	
0Ch	23:0	psv0	[EMC] First input PSV index. Must be aligned to num_psv (for example, if num_psv=2, PSV0 should be even number). The PSV-number for the other PSVs is PSV0+1, PSV0+2 etc.	

Table 1834 - CREATE_PSV Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch
								psv0_index																								10h
								psv1_index																								14h
								psv2_index																								18h
								psv3_index																								1Ch

Table 1835 - CREATE_PSV Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h	23:0	psv0_index	Index of psv0	
14h	23:0	psv1_index	Index of psv1	
18h	23:0	psv2_index	Index of psv2	

Table 1835 - CREATE_PSV Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
1Ch	23:0	psv3_index	Index of psv3	

29.7.2 DESTROY_PSV

DESTROY_PSV destroys the PSV whose number is in the input parameter field.

Table 1836 - DESTROY_PSV Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																psvn																08h
																																0Ch

Table 1837 - DESTROY_PSV Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	psvn	PSV number	
0Ch	31:0	reserved		

Table 1838 - DESTROY_PSV Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1839 - DESTROY_PSV Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.7.3 QUERY_PSV – Query PSV

Query PSV returns the PSV context, including the extended section.

Table 1840 - QUERY_PSV Input Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								psv_number																								08h
																																0Ch

Table 1841 - QUERY_PSV Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	psv_number		
0Ch	31:0	reserved		

29.8 EQ Commands

EQs are created through the CREATE_EQ command. To create an EQ and map events to it, software must prepare the EQ parameters in the software format in the input mailbox and issue the CREATE_EQ command. HW returns EQ handle to be used by SW for further reference to this EQ.

Association of completion events with EQs is controlled by the CQ configuration.

For completion events, EQEs are reported according to the configured EQ number in the CQ.

Events that are not mapped to any EQ are not reported by hardware. To query the EQ state and parameters, software can use the command `QUERY_EQ`. This command retrieves the EQ context and stores it in the output mailbox in software format. The command has no effect on EQ state or EQ execution.

To destroy an EQ, software should execute the `DESTROY_EQ` command. After this command is executed, the last image (contents) of the EQC is stored in the output mailbox, and the EQ becomes invalid. It is the responsibility of software to disassociate all events previously mapped to this EQ prior to executing the `DESTROY_EQ` command.

Note: It is the responsibility of software to map events to existing and configured EQs and not to destroy an EQ while events are still mapped to the EQ.

The commands listed in Table 1842 are used for setting up and maintaining EQs.

Table 1842 - EQ Commands Overview

Mnemonic	Command	Description	Reference
<code>CREATE_EQ</code>	Create EQ	Creates new EQ	Section 29.8.1, on page 1880
<code>DESTROY_EQ</code>	Destroy EQ	Closes the EQ and invalidates the EQC entry from hardware to software	Section 29.8.2, on page 1883
<code>QUERY_EQ</code>	Query EQ	Retrieves a snapshot of the current EQC entry	Section 29.8.3, on page 1884
<code>GEN_EQE</code>	Generate Event Queue Entry	Generate an Event Queue Entry	Section 29.8.4, on page 1886

29.8.1 `CREATE_EQ` – Create EQ

`CREATE_EQ` command creates new EQ. The command takes the EQC entry from the input structure and uses it for a new EQ. The command also maps events to the EQ. For each bit set in the bitmask, the corresponding event group will be mapped to the EQ. If the event group was previously mapped to another EQ, it will be remapped to the new EQ.

Event groups that are not mapped to any EQ are not reported by the hardware. Destroying the EQ will cause all events mapped to this EQ to be unmapped.

Note that there will be no Command Interface Completion Event on `DESTROY_EQ` and `CREATE_EQ` for the EQ that this event is mapped to. Additionally, if `CREATE_EQ` fails, the events trying map to it will not remain unmapped.

Table 1843 - `CREATE_EQ` Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																								input_eqn				08h				

Table 1843 - CREATE_EQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch
eq_context (See Table 244, “Event Queue Context Layout,” on page 505)																																10h-4Ch
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
event_bitmask_63_0																																58h-5Ch
event_bitmask_127_64																																60h-64h
event_bitmask_191_128																																68h-6Ch
event_bitmask_255_192																																70h-74h
																																78h-10Ch
pas[0]																																110h-114h
pas[1]																																118h-11Ch
...																																...

Table 1844 - CREATE_EQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	input_eqn	Input eqn	

Table 1844 - CREATE_EQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-4Ch	512	eq_context (See Table 244, “Event Queue Context Layout,” on page 505)		
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. . Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803).	
58h-5Ch	64	event_bitmask_63_0	See Table 175, “Event Type and Coding,” on page 477 Bitmask[i] is related to the event with event-type=i etc. bit-mask[0] is reserved.	
60h-64h	64	event_bitmask_127_64	See Table 175, “Event Type and Coding,” on page 477 Bitmask[i] is related to the event with event-type=i etc. bit-mask[0] is reserved.	
68h-6Ch	64	event_bitmask_191_128	See Table 175, “Event Type and Coding,” on page 477 Bitmask[i] is related to the event with event-type=i etc. bit-mask[0] is reserved.	
70h-74h	64	event_bitmask_255_192	See Table 175, “Event Type and Coding,” on page 477 Bitmask[i] is related to the event with event-type=i etc. bit-mask[0] is reserved.	
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)		

Table 1845 - CREATE_EQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																								eq_number								08h
																																0Ch

Table 1846 - CREATE_EQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 1846 - CREATE_EQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	7:0	eq_number	eq number	
0Ch	31:0	reserved		

The command delivers the pages for the EQ. All pages should be initialized by SW prior to posting this command such that the ‘ownership’ on all EQEs will be equal to ‘1’ (initial HW ownership).

29.8.2 DESTROY_EQ – Destroy EQ

The DESTROY_EQ command closes the EQ and invalidates the EQC entry from hardware to software. It is the responsibility of the software to unmap all the events, which were previously mapped to the EQ, prior to issuing the DESTROY_EQ command.

Note that there will be no event on destroy EQ for the EQ that the event is mapped to it. Also, if DESTROY_EQ fails, the events are still mapped to the EQ and cannot be mapped to another EQ.

Table 1847 - DESTROY_EQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																							eq_number						08h			
																																0Ch

Table 1848 - DESTROY_EQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	eq_number	eq_number	
0Ch	31:0	reserved		

Table 1849 - DESTROY_EQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h

Table 1849 - DESTROY_EQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																04h
																																08h-0Ch

Table 1850 - DESTROY_EQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.8.3 QUERY_EQ – Query EQ

The QUERY_EQ command retrieves a snapshot of the current EQC entry. The command stores the snapshot in the output mailbox in the software format. Note that the EQC state and values are not affected by the QUERY_EQ command.

The QUERY_EQ command is for debug purposes only.

Table 1851 - QUERY_EQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
																							eq_number					08h					
																																	0Ch

Table 1852 - QUERY_EQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	eq_number	EQ number	

Table 1852 - QUERY_EQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	reserved		

Table 1853 - QUERY_EQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																004h				
																																08-0Ch				
eq_context (See Table 244, “Event Queue Context Layout,” on page 505)																																10h-4Ch				
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h				
event_bitmask																																58h-5Ch				
																																60h-10Ch				
pas[0]																																110h-114h				
pas[1]																																118h-11Ch				
...																																...				

Table 1854 - QUERY_EQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 1854 - QUERY_EQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-4Ch	512	eq_context (See Table 244, “Event Queue Context Layout,” on page 505)		
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803).	
58h-5Ch	64	event_bitmask		
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)		

29.8.4 GEN_EQE – Generate Event Queue Entry

The GEN_EQE command generates EQE on the specified EQ.

Table 1855 - GEN_EQE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								eq_number				08h				
																																0Ch
eqe																																10h-4Ch

Table 1856 - GEN_EQE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	eq_number	eq_number on which we generate the EQE	

Table 1856 - GEN_EQE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-4Ch	512	eqe	EQE to generate	

Table 1857 - GEN_EQE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1858 - GEN_EQE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.9 CQ Commands

CQs are created using the CREATE_CQ command. Software prepares the CQ in the software format, places it in the mailbox and issues the command. HW returns CQ handle to be used for further reference. To destroy a CQ, software should issue the DESTROY_CQ command on the CQ.

The QUERY_CQ command is used only for debugging and provides a snapshot of the current CQ state.

The MODIFY_CQ command is used for resizing a CQ, modifying the CQ moderation parameters and changing the CQ to EQ mapping.

The following commands are used for setting up and maintaining CQs.

Table 1859 - CQ Commands Overview

Mnemonic	Command	Description	Reference
CREATE_CQ	Create CQ	Creates new CQ	Section 29.9.1, on page 1888

Table 1859 - CQ Commands Overview (Continued)

Mnemonic	Command	Description	Reference
DESTROY_CQ	Destroy CQ	Destroys CQ	Section 29.9.2, on page 1890
QUERY_CQ	Query CQ	Retrieves a snapshot of the current CQC entry	Section 29.9.3, on page 1891
MODIFY_CQ	Modify CQ	Modify CQ parameters (resize, change moderation parameters, modify CQ to EQ mapping)	Section 29.9.4, on page 1894

29.9.1 CREATE_CQ – Create Completion Queue

CREATE_CQ command creates new CQ.

The command also delivers the pages for the CQ. All pages should be initialized by SW prior to posting this command such that:

- The OPCODE field on all CQEs will be equal to 0xF (invalid CQE).
- The owner bit =Hardware.

Table 1860 - CREATE_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								input_cqn																								08h
																																0Ch
cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)																																10h-4Ch
e_mtt_pointer_or_cq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
cq_umem_id																																58h
cq_umem_valid																																5Ch

Table 1860 - CREATE_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																60h-10Ch
pas[0]																																110h-114h
pas[1]																																118h-11Ch
...																																...

Table 1861 - CREATE_CQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	input_cqn		
10h-4Ch	512	cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)		
50h-54h	64	e_mtt_pointer_or_cq_umem_offset (See Table 1802, “e_mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803). If ~~UID~~ UID!=0cq_umem_valid==1, then indicates umem offset of the CQ	
58h	31:0	cq_umem_id	UMEM id of the CQ. Valid only cq_umem_valid==1.	
5Ch	31	cq_umem_valid	Indicates if cq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	

Table 1861 - CREATE_CQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)	Valid only when cq_umem_valid==0.	

Table 1862 - CREATE_CQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																004h				
								cqn																												08h
																																0Ch				

Table 1863 - CREATE_CQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	cq_n	CQ number	
0Ch	31:0	reserved		

29.9.2 DESTROY_CQ – Destroy CQ

The DESTROY_CQ command destroys a CQ.

Table 1864 - DESTROY_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								cq_n																								08h
																																0Ch

Table 1865 - DESTROY_CQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	cqn	CQ number	
0Ch	31:0	reserved		

Table 1866 - DESTROY_CQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1867 - DESTROY_CQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.9.3 QUERY_CQ – Query CQ

The QUERY_CQ command retrieves a snapshot of the current CQC entry. The command stores the snapshot in the output mailbox in the software format. Note that the CQC state and values are not affected by the QUERY_CQ command.

The QUERY_CQ command is for debug purposes only.

Table 1868 - QUERY_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h

Table 1868 - QUERY_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																op_mod												04h				
								cqn																				08h				
																																0Ch

Table 1869 - QUERY_CQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	cqn	CQ number	
0Ch	31:0	reserved		

Table 1870 - QUERY_CQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08-0Ch
cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)																																10h-4Ch
e_mtt_pointer_or_cq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
cq_umem_id																																58h
cq_umem_valid																																5Ch

Table 1870 - QUERY_CQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																60h-10Ch
pas[0]																																110h-114h
pas[1]																																118h-11Ch
...																																...

Table 1871 - QUERY_CQ Output Structure Layout

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)		
50h-54h	64	e_mtt_pointer_or_cq_umem_offset (See Table 1802, “e_mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803). If cq_umem_valid==1, then indicates umem offset of the CQ	
58h	31:0	cq_umem_id	UMEM id of the CQ. Valid only cq_umem_valid==1 .	
5Ch	31	cq_umem_valid	Indicates if cq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	
110h-...	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)		

29.9.4 MODIFY_CQ – Modify CQ Parameters

The MODIFY_CQ command is used for resizing an existing CQ, or changing the CQ to EQ mapping and modifying the CQ context after creation (such as moderation parameters). The field to be modified is selected using the *field_select* bitmask, and the relevant fields passed as part of CQC.

Table 1872 - MODIFY_CQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								cqn																								08h
modify_field_select_or_resize_field_select																																0Ch
cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)																																10h-4Ch
cq_umem_offset																																50h-54h
cq_umem_id																																58h
cq_umem_valid																																5Ch
																																60h-10Ch
	pas[0]																															110h-114h
	pas[1]																															118h-11Ch
	...																															...

Table 1873 - MODIFY_CQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: modify_cq 1: resize_cq - See Table 8.19.6, “Resizing a CQ,” on page 453	
08h	23:0	cqn		
0Ch	31:0	modify_field_select_or_resize_field_select	Specifies which fields of CQ context is selected. When op_mod = 0: modify_field_select. See Table 1874, “MODIFY_FIELD_SELECT Structure Layout,” on page 1895. When op_mod = 1: resize_field_select. See Table 1876, “RESIZE_FIELD_SELECT Structure Layout,” on page 1896.	
10h-4Ch	512	cq_context (See Table 171, “Completion Queue Context Layout,” on page 467)		
50h-54h	64	cq_umem_offset	Valid only if cq_umem_valid==1, then indicates umem offset of the CQ	
58h	31:0	cq_umem_id	UMEM id of the CQ. Valid only cq_umem_valid==1 .	
5Ch	31	cq_umem_valid	Indicates if cq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	
110h-...	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)	Valid only for resize_cq (op_mode=1) and when cq_umem_valid==0 .	

Table 1874 - MODIFY_FIELD_SELECT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_field_select																																00h

Table 1875 - MODIFY_FIELD_SELECT Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	modify_field_select	When op_mod=0 (modifying CQ fields) Selects fields to be modified Bit 0: cq_period - this field can be modified only if cq_moderation in QUERY_HCA_CAP is set. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667. Bit 1: cq_max_count - this field can be modified only if cq_moderation in QUERY_HCA_CAP is set. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667 Bit 2: oi - this field can be modified only if cq_oi in QUERY_HCA_CAP is set. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667. Bit 3: c_eqn - this field can be modified only if HCA_CAP.cq_eq_remap is set. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667. Bit 4: cq_period_mode - this field can be modified only if HCA_CAP.cq_period_mode_modify==1. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667 Bit 5: status - this field can be modified only if HCA_CAP.cq_invalidate==1, and SW allowed to modify to CQ_INVALID only. See Section 29.3.4, “QUERY_HCA_CAP – Query Device Capabilities,” on page 1667	

Table 1876 - RESIZE_FIELD_SELECT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
resize_field_select																																00h

Table 1877 - RESIZE_FIELD_SELECT Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	resize_field_select	When op_mod=1(Resize_Cq) Bit 0: log_cq_size - Supported only when cq_umem_valid==0 Bit 1: page_offset - Supported only when cq_umem_valid==0 Bit 2: log_page_size - Supported only when cq_umem_valid==0 Bit 3: umem - umem_id,umem_offset fields and log_cq_size. Supported only when cq_umem_valid==1.	

Table 1878 - MODIFY_CQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																004h				
																																08h-0Ch				

Table 1879 - MODIFY_CQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.10 QP Commands

The commands listed in Table 1880, “QP Commands” on page 1898 modify QP state and control ownership of QPs. All these commands share the same basic input/output format, which is a QP context (see Table 19, “QP Context Structure Layout,” on page 258). For each command, only the relevant fields of the QP format need to be filled; all other fields are ignored by hardware. When optional parameters may be provided, a mask will indicate which of the parameters are supplied by software.

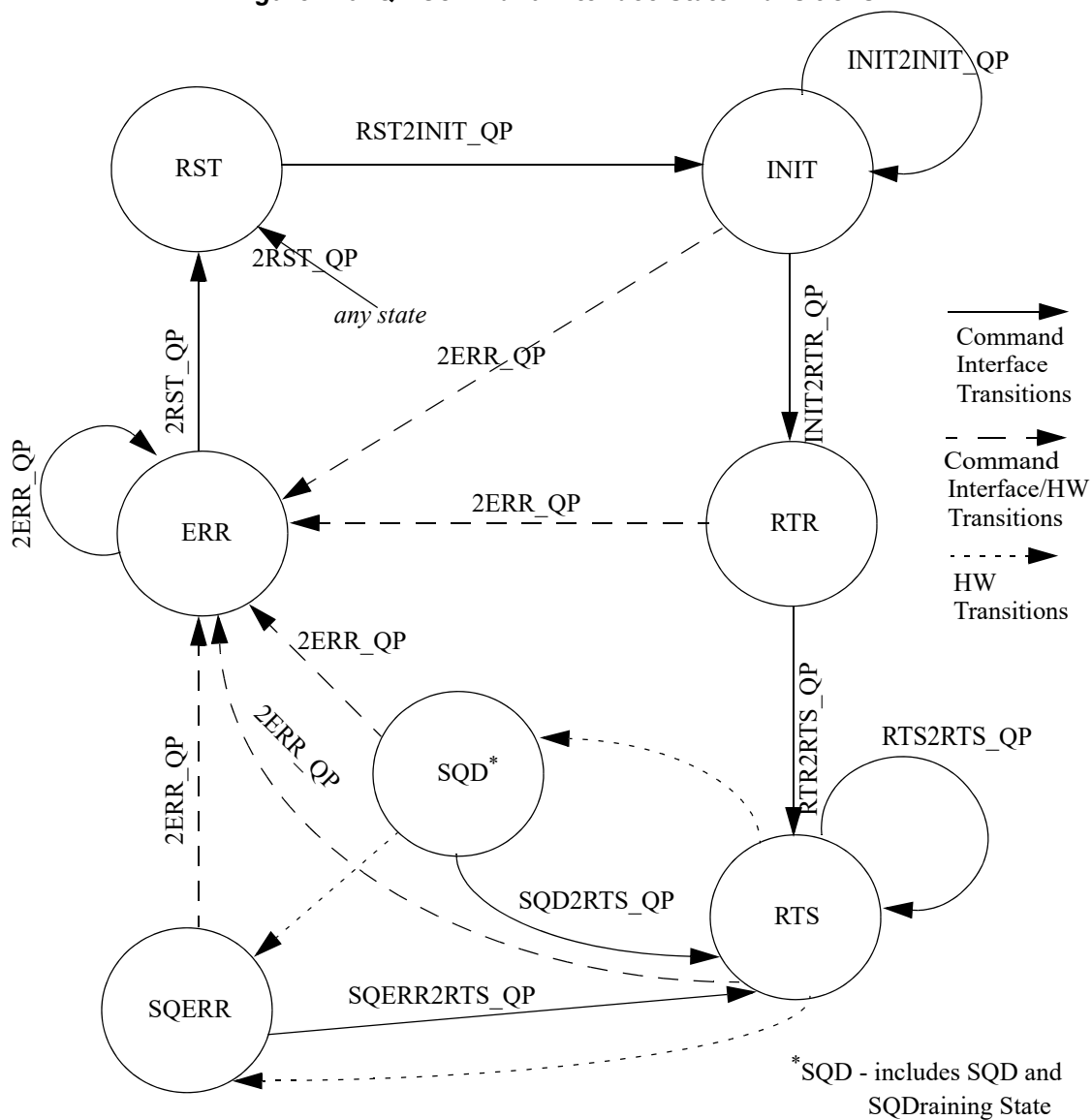
The QUERY_QP command generates a snapshot of current QP attributes and does not affect the current QP state.

Figure 116 summarizes all QP state transitions triggered by the command interface and the commands that trigger these transitions. It is the responsibility of software to activate QP state transitions only at the appropriate states. However, hardware may transition a QP asynchronously into the ERR or SQERR states. When a state transition is requested through the command interface and the QP is in the wrong state, the command will terminate with error status.

The SQD state (Send Queue Drained) has an interim state, which does not appear in the diagram. When software requests a transition to the SQD state, QP will move to the (interim) SQ Draining state and will finish processing of the WQEs currently processed on the SQ. Once finished, QP will move to the SQD state. The SQ Draining state is a special state from the perspective of software. Software can only move from this state to RST or ERR state. It is not possible to move from SQ Draining state into RTS state. To move to RTS, software should wait for the QP to reach SQD State.

State transitions of special QPs are activated on the QPs that are mapped to the special QPs. See “QP Command Interface State Transitions” on page 1898 for additional information.

Figure 116: QP Command Interface State Transitions



The following commands are used to set up and maintain QP contexts.

Table 1880 - QP Commands

Mnemonic	Command	Description	Reference
CREATE_QP	Allocate QP	Allocates QP and returns QP handle	Section 29.10.1, on page 1899
DESTROY_QP	Destroy QP	Destroys QP	Section 29.10.2, on page 1904
RST2INIT_QP	Reset to Init QP	Transition QP from reset to INIT	Section 29.10.3, on page 1905

Table 1880 - QP Commands (Continued)

Mnemonic	Command	Description	Reference
INIT2INIT_QP	Init to Init QP	Modify QP attributes for QP in INIT	Section 29.10.4, on page 1907
INIT2RTR_QP	Init to RTR QP	Transition QP from INIT to RTR	Section 29.10.5, on page 1909
RTR2RTS_QP	RTR to RTS QP	Transition QP from RTR to RTS	Section 29.10.6, on page 1914
RTS2RTS_QP	RTS to RTS QP	Modify QP attributes for QP in RTS	Section 29.10.7, on page 1917
SQERR2RTS_QP	SQERR to RTS QP	Transition QP from SQERR to RTS	Section 29.10.8, on page 1922
2ERR_QP	To Error QP	Transition QP to ERR	Section 29.10.9, on page 1924
2RST_QP	To Reset QP	Transition QP to reset	Section 29.10.10, on page 1925
QUERY_QP	Query QP	Query QP	Section 29.10.11, on page 1927
SQD2RTS	SQD to RTS	Transition QP from SQD to RTS	Section 29.10.12, on page 1929
SUSPEND_QP	Suspend QP	Suspends the operation of a QP	Section 29.10.13, on page 1931
RESUME_QP	Resume QP	Resumes the operation of a QP	Section 29.10.14, on page 1932
SET_RDB	Set RDB Context	Set RDB Context	Section 29.10.15, on page 1933
QUERY_RDB	Query RDB Context	Query RDB Context	Section 29.10.16, on page 1935

29.10.1 CREATE_QP - Create new QP

CREATE_QP command creates a new QP and places it in RST state. The command takes the QPC entry from the input mailbox and uses it for new QP. CREATE_QP also transfers the physical pages of the WQ buffer. The first set of PF physical buffers are in the input mailbox. If there is not enough place there, more pages are transferred using the link list started with the “next pointer” in the inbox.

Table 1881 - CREATE_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
								input_qpn																				08h				
																																0Ch
opt_param_mask																																10h

Table 1881 - CREATE_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ece																																14h
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																																18h-FCb
e_mtt_pointer_or_wq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																100h-104h
wq_umem_id																																108hh
wq_umem valid																																10Ch
pas[0]																																110h-114h
pas[1]																																118h-11Ch
...																																...

Table 1882 - CREATE_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	input_qpn	qpn of the created qp.0 means reserved and qpn is returned by device. Supported only when HCA_CAP.qp_by_name==1. .	

Table 1882 - CREATE_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	opt_param_mask	Should be 0 as there are no optional parameters.	
<u>14h</u>	<u>31:0</u>	<u>ece</u>	<u>Enhanced Connection Establishment provided by device.</u> <u>0 means invalid.</u> <u>Supported when HCA_CAP.ece = 1</u>	
18h-FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h-104h	64	e_mtt_pointer_or_wq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803) Pointer to MTTs in the ICM. Reserved when rq_type=no_rqw and sq_type=no_sq=1 In case wq_umem_valid==1, indicates offset in allocated umem id.	
108h	31:0	wq_umem_id	UMEM ID of the wq. valid only when wq_umem_valid==1.	
10Ch	31	wq_umem_valid	Indicates if wq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset are valid	
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)	Physical addresses of the WQ buffers valid only when wq_umem_valid==0.	

Table 1883 - CREATE_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																												04h				
								qpn																							08h	
<u>ece</u>																												0Ch				

Table 1884 - CREATE_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	qpn	QP number	
0Ch	31:0	ece		

For the CREATE_QP command, the following parameters in *qpc_data* must be valid (parameters marked with ‘n’ are not required for the specified service type).

Table 1885 - CREATE_QP Required Parameters

Category	Parameters	RC	UC	UD	DCI	Comments MLX
Fields from QPC	st					
	pd					
	log_rq_size				n	
	log_rq_stride				n	
	log_sq_size					
	no_sq					
	rlky					
	uar_page					
	log_page_size					
	fre					n
	cqn_snd					
	page_offset					
	cqn_rcv					
	dbr_addr					
	rq_type					
	srqn/rmpn					
	cs_req		n	n		n
	cs_res				n	

Table 1885 - CREATE_QP Required Parameters

Category	Parameters	RC	UC	UD	DCI	Comments MLX
	cgs	n	n		n	n
	wq_signature					
	cd_master					
	cd_slave_send					
	cd_slave_receive				n	
	atomic_mode		n	n	n	n
	atomic_like_write_en				n	
	latency_sensitive					
	Primary Address Path: free_ar	n			n	
	block_lb_mc	n	n		n	
	user_index					
	drain_sigerr					
	ulp_stateless_offload_mode	n	n		n	n
	dual_write		n	n	n	n
	full_handshake	n	n	n		n
	cnak_reverse_sl	n	n	n		n
	end_padding_mode					
	offload_type		n	n	n	
	req_e2e_credit_mode		n	n	n	n
	log_num_dci_stream_channels	n	n	n		
	log_num_dci_errored_streams	n	n	n		

opt_param_mask must be zero, since the command does not support optional parameters

29.10.2 DESTROY_QP – Destroying a QP

The DESTROY_QP command destroys a QP and enables the device to release the related resources.

Table 1886 - DESTROY_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
									qp_n																			08h				
																																0Ch

Table 1887 - DESTROY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qp_n	qp_number	

Table 1888 - DESTROY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1889 - DESTROY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 1889 - DESTROY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		

29.10.3 RST2INIT_QP - Reset to Init QP

The RST2INIT_QP command transitions a QP to init state. The command enables posting WQEs and on the RQ, but incoming packets on the QP will be silently discarded.

Table 1890 - RST2INIT_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																qp_n																08h
																																0Ch
																opt_param_mask																10h
																ece																14h
																qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																18h-FCh
																																100h-10Ch

Table 1891 - RST2INIT_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qp_n	QP number	

Table 1891 - RST2INIT_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
<u>14h</u>	<u>31:0</u>	<u>ece</u>	<u>Enhanced Connection Establishment provided by device. 0 means invalid. Supported when HCA_CAP.ece = 1</u>	
18h-FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h-10Ch	128	reserved		

Table 1892 - RST2INIT_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h
<u>ece</u>																																0Ch

Table 1893 - RST2INIT_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:0	ece		

Table 1894 - RST2INIT_QP Required Parameters

Valid Input Parameters	RC	UC	UD	DCI	Comments MLX
q_key	n	n		n	

Table 1894 - RST2INIT_QP Required Parameters (Continued)

Valid Input Parameters	RC	UC	UD	DCI	Comments MLX
DC_access_key	n	n	n		n
Primary Address Path: Physical port					
Primary Address Path: pkey_index					For IB port only
rwe			n	n	n
rae, atomic_mode		n	n	n	n
rre		n	n	n	n
pm_state					
counter_set_id					
lag_tx_port_affinity					

29.10.4INIT2INIT_QP – Init to Init QP

The INIT2INIT_QP command performs no state transition on the QP. The command allows replacing some of the QP parameters.

There are no required parameters for the INIT2INIT_QP command. The following parameters are optional. When software specifies an optional parameter, it should set the corresponding bit in *opt_param_mask*.

Table 1895 - INIT2INIT_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
									qpn																		08h					
																																0Ch
opt_param_mask																																10h
																																14h
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																																18h-10Ch
																																100h-10Ch

Table 1896 - INIT2INIT_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
18h- FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h- 10Ch	128	reserved		

Table 1897 - INIT2INIT_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				004h
																																				08h-0Ch

Table 1898 - INIT2INIT_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 1899 - INIT2INIT_QP Optional Parameters Mask

Bit	Valid Input Params	RC	UC	UD	DCI	Comments MLX
16	Primary Address Path: Physical port					
21	dc_access_key	n	n	n		n
5	q_key	n	n		n	
4	Primary Address Path: pkey_index					For IB port only
3	rwe			n	n	n
2	rae, atomic_mode		n	n	n	n
1	rre		n	n	n	n
15	lag_tx_port_affinity Supported only when HCA_ CAP.init2_lag_tx_port_affin- ity==1					

29.10.5 INIT2RTR_QP – Init to RTR QP

The INIT2RTR_QP command activates the RQ. Once the command has been completed, the RQ will respond to incoming requests.

Refer to Table 1904, “INIT2RTR_QP Required Parameters,” on page 1911 and Table 1905, “INIT2RTR_QP Optional Parameters Mask,” on page 1913 for required and optional parameters that should be supplied.

Table 1900 - INIT2RTR_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
								qpn																				08h				
																																0Ch
opt_param_mask																																10h
eee																																14h
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																																18h-FCh

Table 1900 - INIT2RTR_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																100h-10Ch

Table 1901 - INIT2RTR_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
<u>14h</u>	<u>31:0</u>	<u>ece</u>	<u>Enhanced Connection Establishment provided by device. 0 means invalid. Supported when HCA_CAP.ece = 1</u>	
18h- FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h- 10Ch	128	reserved		

Table 1902 - INIT2RTR_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h

Table 1902 - INIT2RTR_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 0Ch
ece																																0Ch

Table 1903 - INIT2RTR_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:0	ece		

Table 1904 - INIT2RTR_QP Required Parameters

Category	Parameters	RC	UC	UD	DCI	Comments MLX
All fields from QPC:	mtu			n		n
	log_msg_max			n		n
	remote_qpn / wire_qpn			n	n	n
	log_rra_max		n	n	n	n
	next_rev_psn			n	n	n
	min_rnr_nak		n	n	n	n
	packet_pac- ing_rate_limit_in- dex					
	deth_sqpn	n	n		n	n
	initiator_src_dct	n	n	n		
Address path from QPC:	FL			n	n	[TBD]??F or IB Only-n
	grh			n	n	For IB port only-n
	mlid			n	n	For IB port only-n

Table 1904 - INIT2RTR_QP Required Parameters (Continued)

Category	Parameters	RC	UC	UD	DCI	Comments MLX
	rlid			n	n	For IB port only- n
	rmac			n	n	For RoCE Only- n
	src_addr_index			n	n	n
	stat_rate			n	n	n
	hop_limit			n	n	n
	tclass			n For IB port only	n For IB port only	n
	flow_label			n	n	n
	rgid/rip			n	n	n
	sl			n		For IB port only
	eth_prio					- n
	DEI/CFI					For RoCE Only- n
	udp_sport			n	n	For RoCE Only- n
	dscp					For RoCE Only- n
	ecn					For RoCE Only- n
	f_eth_eup	n	n			For RoCE Only- n
	f_dscp	n	n			For RoCE Only- n
	f_ecn	n	n			For RoCE Only- n
	multi_path		n	n	n	n

The following parameters are optional. When software specifies an optional parameter, it should set the corresponding bit in *opt_param_mask* (parameters marked with ‘n’ are invalid for the specific transport service).

Table 1905 - INIT2RTR_QP Optional Parameters Mask

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
21	DC_access_key	n	n	n		n
5	q_key	n	n		n	
4	Primary Address Path: pkey_index					
3	rwe			n	n	n
2	rae, atomic_mode		n	n	n	n
1	rre		n	n	n	n
11	drain_sigerr (Supported only when HCA_- CAP.init2rtr_drain_sigerr ==1)					
0	<i>Alternate Address Path:</i> pkey_index grh mlid rlid ack_timeout log_rtm src_addr_index stat_rate hop_limit tclass flow_label rgid sl FL Physical port prio (for RoCE only) DEI/CFI (for RoCE only) dscp (for RoCE only) ecn(for RoCE only) rmac(for RoCE only)			n	n	[Internal] need to update relavent fields for ROCEn

Table 1905 - INIT2RTR_QP Optional Parameters Mask (Continued)

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
15	lag_tx_port_affinity Supported only when HCA_CAP.init2_lag_tx- _port_affinity==1					
31	sr_enable		n	n		

29.10.6RTR2RTS_QP –RTR to RTS QP

The RTR2RTS_QP command activates the SQ. Once the command has been completed, the SQ will start processing DoorBells, and WQEs posted to the SQ will be executed by the device.

Refer to Table 1910, “RTR2RTS_QP Required Parameters,” on page 1916 and Table 1911, “RTR2RTS_QP Optional Parameters Mask,” on page 1916 for required and optional parameters that should be supplied.

When loading an alternate path, or if an alternate path was loaded while the QP was in the INIT2RTR_QP state transition, the **pm_state** should be initialized to the REARM state, in order to trigger the arming mechanism of the automatic path migration. Note that execution time of the command will be longer if **pm_state** is set to REARM. If **pm_state** is modified by a command, that is, the optional parameter bit that corresponds to the **pm_state** is set, an AutomaticPathMigration FSM transition is considered to be requested.

For the RTR2RTS_QP command, the following parameters in **qpc_data** must be valid (parameters marked with ‘n’ are not required for the specified service type).

Table 1906 - RTR2RTS_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
									qpn																		08h					
																																0Ch
opt_param_mask																																10h
ece																																14h
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																																18h-FCh
																																100h-10Ch

Table 1907 - RTR2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
~~14h~~ 14h	~~31:0~~ 31:0	~~ece~~	<u>Enhanced Connection Establishment provided by device. 0 means invalid. Supported when HCA_CAP.ece = 1</u>	
18h- FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h- 10Ch	128	reserved		

Table 1908 - RTR2RTS_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h
<u>ece</u>																																0Ch

Table 1909 - RTR2RTS_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:0	ece		

Table 1910 - RTR2RTS_QP Required Parameters

Category	Parameters	R C	U C	U D	DCI	Comments MLX
All fields from QPC:	log_ack_req_freq		n	n		n
	log_sra_max		n	n		n
	next_send_psn					n
	retry_count		n	n		n
	extended_retry_count_valid		n	n		
	extended_retry_count		n	n		
	rnr_retry		n	n		n
Address path from QPC:	ack_timeout		n	n		n
	log_rtm		n	n		n

The following parameters are optional. When software specifies an optional parameter, it should set the corresponding bit in *opt_param_mask* (parameters marked with ‘n’ are invalid for the specific transport service).

Table 1911 - RTR2RTS_QP Optional Parameters Mask

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
21	DC_access_key	n	n	n		n
10	pm_state			n		n
6	min_rnr_nak		n	n	n	n
5	q_key	n	n		n	n

Table 1911 - RTR2RTS_QP Optional Parameters Mask (Continued)

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
3	rwe			n	n	n
2	rae, atomic_mode		n	n	n	n
22	packet_pacing_rate_limit_index					
24	offload_type		n	n	n	Supports only NONE <==> NVMF
0	Alternate Address Path: pkey_index grh mlid rlid ack_timeout log_rtm src_addr_index stat_rate hop_limit tclass flow_label rgid sl FL rmac eth_prio DEI/CFI dscp ecn			n	n	[Internal] need to update relevant fields for ROCEn

29.10.7RTS2RTS_QP – RTS to RTS QP

The RTS2RTS_QP command, which is not a state transition, allows modification of some of the parameters of the QP without interfering with their normal operation.

Refer to Table 1916, “RTS2RTS_QP Optional Parameters Mask,” on page 1919 for optional parameters that should be supplied.

While the command is executed, both the SQ and RQ will continue receiving DoorBells and processing WQEs as well as incoming requests from IB. RTS2RTS_QP enables restricting and enabling RDMA operations, altering the QKey, loading an alternate path and changing the minimum RNR Nak time-out.

Note that if disable RDMA is requested while incoming RDMAs are in process, when the disable will take effect is not determined.

Note that execution time of the command will be longer if path migration is requested, or an alternate path is logged (*pm_state* different from MIGRATED). If *pm_state* is modified by a command, that is, the optional parameter bit that corresponds to the *pm_state* is set, an AutomaticPathMigration FSM transition is considered to be requested.

Table 1912 - RTS2RTS_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																qp_n																08h
																																0Ch
																opt_param_mask																10h
																ece																14h
																qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																18h-FCh
																dci_stream_channel_id																100h
																																104h-10Ch

Table 1913 - RTS2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod	0x0: QP_ACCESS 0x1: STREAM_CHANNEL_ACCESS	
08h	23:0	qp_n	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	

Table 1913 - RTS2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
14h	31:0	<u>ece</u>	<u>Enhanced Connection Establishment provided by device.</u> <u>0 means invalid.</u> <u>Supported when HCA_CAP.ece = 1</u>	
18h- FCh	1856	qpc_data (See Table 19, “QP Con- text Structure Layout,” on page 258)	QPC data.	
100h	15:0	dci_stream_channel_id	DCI Stream Channel Identifier. Valid only when op_mod==STREAM_CHANNEL	
104h- 10Ch	128	reserved		

Table 1914 - RTS2RTS_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h
<u>ece</u>																																0Ch

Table 1915 - RTS2RTS_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:0	ece		

The following parameters are optional. When software specifies an optional parameter, it should set the corresponding bit in *opt_param_mask* (parameters marked with ‘n’ are invalid for the specific transport service):

Table 1916 - RTS2RTS_QP Optional Parameters Mask

Bit	Valid Input Param	RC	UC	UD	DCI	Comments ^{MLX}
19	cqn_rcv	n	n			n

Table 1916 - RTS2RTS_QP Optional Parameters Mask (Continued)

Bit	Valid Input Param	RC	UC	UD	DCI	CommentsMLX
18	~~srqn~~ rmpn_xrqn This field is supported only when rq_type==1 and HCA_CAP.rts2rts_qp_rmp==1		n	n	n	The modify of this field is best effort, and command will fail (BAD_RESOURCE_STATE) and no modifications done in the following cases: QP is in the middle of receiving a message or the original RMP is XRQ. During this command execution, incoming packets to this QP may be dropped, and no RNR_NACK will be sent for them. SW can be aware of moving to new rq by: - New XRQ use new CQ. Note 1: Can't update APM at the same command. To update APM, another call to RTS2RTS_QP is required. Note 2: Can't be during transmit.
10	pm_state			n		n
21	DC_access_key	n	n	n		n
6	min_rnr_nak		n	n	n	n
5	q_key	n	n		n	n
3	rwe			n	n	n
2	rae, atomic_mode		n	n	n	n
1	rre		n	n	n	n
22	packet_pacing_rate_limit_index					

Table 1916 - RTS2RTS_QP Optional Parameters Mask (Continued)

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
23	Primary Address Path: src_addr_index is supported only for RoCE when RoCE_CAP.rts2rts_src_addr_index_for_vlan_valid_vlan_id==1					
24	offload_type		n	n	n	Supports only NONE <==> NVMF
20	Primary Address Path: eth_prio supported only for RoCE when RoCE_CAP.rts2rts_primary_eth_prio==1. sl modify supported only for IB when HCA_CAP.rts2rts_primary_sl==1					
0	Alternate Address Path: pkey_index grh mlid rlid ack_timeout log_rtm src_addr_index stat_rate hop_limit tclass flow_label rgid sl FL rmac prio DEI/CFI dscp ecn udp_sport			n	n	[Internal] need to update relevant fields for ROCE n

Table 1916 - RTS2RTS_QP Optional Parameters Mask (Continued)

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
25	counter_set_id Supported only when HCA_- CAP.rts2rts_qp_counters_ set_id==1					
17	Primary Address Path: dscp and f_dscp is sup- ported only for RoCE when RoCE_- CAP.rts2rts_qp_dscp==1					

29.10.8SQERR2RTS_QP – SQErr to RTS QP

Then SQERR2RTS_QP command enables recovering an SQ from SQ error. Activating the command transitions the SQ into RTS mode.

Refer to Table 1921, “SQERR2RTS_QP Optional Parameters Mask,” on page 1924 for optional parameters that should be supplied.

Table 1917 - SQERR2RTS_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
								qpn																08h								
																																0Ch
opt_param_mask																																10h
																																14h
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																																18h-FCh
																																100h-10Ch

Table 1918 - SQERR2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
18h-FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	
100h-10Ch	128	reserved		

Table 1919 - SQERR2RTS_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1920 - SQERR2RTS_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

The following parameters are optional. When software specifies an optional parameter, it should set the corresponding bit in *opt_param_mask* (parameters marked with ‘n’ are invalid for the specific transport service).

Table 1921 - SQERR2RTS_QP Optional Parameters Mask

Bit	Valid Input Param	RC	UC	UD	DCI	Comments MLX
21	DC_access_key	n	n	n		n
6	min_rnr_nak		n	n	n	n
5	q_key	n	n		n	n
3	rwe			n	n	n
2	atomic_mode		n	n		n
1	rre		n	n	n	n

29.10.92ERR_QP – To Error QP

The 2ERR_QP command transitions the QP into the error state from the state it had been in.

Reset to Error state transition is not supported (per IB spec). While the command is executed, some in-process WQEs may complete. Once the QP transitions into the error state, the hardware will flush WQEs that were posted and WQEs that will be posted to both the SQ and the RQ.

When the command is completed successfully, the QP is already in the Error state.

Table 1922 - 2ERR_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																qp_n																08h
																																0Ch

end-on-behalf
function_id=vhca_id

Table 1923 - 2ERR_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	

Table 1924 - 2ERR_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1925 - 2ERR_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.10.102RST_QP – To Reset QP

The 2RST_QP command transitions a QP from any state to reset state.

Once the QP is in reset state, DoorBells are not processed and WQEs are not flushed by hardware.

Table 1926 - 2RST_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
									qpn																		08h					
																																0Ch

Table 1927 - 2RST_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	

Table 1928 - 2RST_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1929 - 2RST_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.10.11QUERY_QP – Query QP

The QUERY_QP command retrieves a snapshot of QP parameters from the hardware. It should be noted that the returned values are snapshots of the RQ and the SQ which are processing WQEs, so it might be that the snapshot is not taken simultaneously from both the SQ and RQ, and that some parameters are obsolete when the command completes.

When the command is completed, the output mailbox will contain the QP parameters in the software format. opt_param_mask will be zero and all the parameters will be filled.

Table 1930 - QUERY_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
									qp_n																		08h					
																																0Ch

Table 1931 - QUERY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qp_n	QP number	

Table 1932 - QUERY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																															004h	
																															08-0Ch	
opt_param_mask																															10h	
ece																															14h	
qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																															18h-FCh	
e_mtt_pointer_or_wq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																															100h-104h	
wq_umem_id																															108h	
wq_umem_valid																															10Ch	
pas[0]																															110h-114h	
pas[1]																															118h-11Ch	
...																															...	

Table 1933 - QUERY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h	31:0	opt_param_mask	Reserved	

Table 1933 - QUERY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
14h	31:0	ece	Enhanced Connection Establishment provided by device. 0 means invalid. Supported when HCA_CAP.ece = 1	
18h-FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data	
100h-104h	64	e_mtt_pointer_or_wq_umem_offset (See Table 1802, “e_mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803) Pointer to MTTs in the ICM. Reserved when rq_type=no_rqw and sq_type=no_sq=1 In case wq_umem_valid==1, indicates offset in allocated umem id.	
108h	31:0	wq_umem_id	UMEM ID of the wq. valid only when wq_umem_valid==1.	
10Ch	31	wq_umem_valid	Indicates if wq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset are valid	
110h-...	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)		

29.10.12SQD2RTS_QP – SQD to RTS QP

The SQD2RTS_QP command activates the send processing on the Send Queue (SQ) which has been drained. Once the command has completed, the SQ resumes processing Send WQEs. While the QP was in the SQD state, newly submitted Send WQEs and unprocessed Send WQEs on the Send Queue were suspended and not executed. Once the SQD2RTS_QP command is executed, the adapter device resumes processing the suspended Send WQEs.

The command has no effect on the Receive Queue (handling DoorBells and processing incoming requests).

Note that the SQD2RTS_QP command will fail if the SQ has not been completely drained. For proper operation, the software should execute this command only after it has received the SQ Drained event.

If the *pm_state* is modified by the command, i.e. the optional parameter bit that corresponds to the *pm_state* parameter is set, an AutomaticPathMigration FSM transition is considered to be requested.

Table 1934 - SQD2RTS_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
																qpn																08h
																																0Ch
																opt_param_mask																10h
																																14h
																qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)																18h-FCh
																																100h-10Ch

Table 1935 - SQD2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	23:0	qpn	QP number	
10h	31:0	opt_param_mask	This field defines which optional parameters are passed. Each bit specifies whether an optional parameter is passed (set) or not (cleared). The opt_param_mask is defined for each QP command.	
18h-FCh	1856	qpc_data (See Table 19, “QP Context Structure Layout,” on page 258)	QPC data.	

Table 1935 - SQD2RTS_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
100h-10Ch	128	reserved		

Table 1936 - SQD2RTS_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1937 - SQD2RTS_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.10.13 SUSPEND_QP - Suspend a QP

The suspend QP command suspends the operation of the given QP. A QP may be suspended in any state. Upon completion of this command, the QP operation is frozen, WQE processing is halted and no CQEs are generated. Nevertheless, all WQEs that were completed prior to suspending the QP generate CQEs if required and will be visible in host memory upon completion of this command. See TBD

Table 1938 - SUSPEND_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								qpn																								08h
																																0Ch

Table 1939 - SUSPEND_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	qp_n	qp number to be suspended	

Table 1940 - SUSPEND_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																08h-0Ch

Table 1941 - SUSPEND_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.10.14 RESUME_QP - Resume a QP

The resume QP command resumes the operation of a given QP. A QP must be in suspended state, either due to a SUSPEND_QP command or due to a newly created QP in suspended state, prior to being resumed. In the case of resuming a newly created QP, SW must post all inflight WQEs in the Send and Receive queues prior to calling RESUME_QP. The RESUME_QP operation reinstates Send queue operation so no additional doorbell is required by SW. See TBD

Table 1942 - RESUME_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
									qpn																								08h
																																0Ch	

Table 1943 - RESUME_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	qpn	qp number to be resumed	

Table 1944 - RESUME_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0C <h></h>

Table 1945 - RESUME_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.10.15 SET_RDB - SET RDB context

The SET_RDB command is used to load the RDB state and populate the extended atomic response of a suspended QP. Both the RDB and the atomic response records should be provided

starting from the last recorded entry up to the most recent entry. RDB records that were never utilized should have the 'invalid' RDB type.

Table 1946 - SET_RDB Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								qpn																								08h
																								rdb_list_size								0Ch
rdb_context[0] (See Table 23, “RDB Context Layout,” on page 278)																																10h-4Ch
rdb_context[1] (See Table 23, “RDB Context Layout,” on page 278)																																50h-8Ch
...																																90h-...

Table 1947 - SET_RDB Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod		
08h	23:0	qpn	qp number.	
0Ch	7:0	rdb_list_size	Size of the rdb list	
10h-...	512	rdb_context[...] (See Table 23, "RDB Context Layout," on page 278)	RDB context list. Size indicated by rdb_list_size.	

Table 1948 - SET_RDB Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 1948 - SET_RDB Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 1949 - SET_RDB Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.10.16 QUERY_RDB - QUERY RDB context

The QUERY_RDB command is used to save the RDB state of a suspended QP.

If QUERY_RDB is invoked on a QP that is not suspended, a BAD_RES_STATE status is returned. Both the RDB and the atomic response records are returned starting from the last recorded entry up to the most recent entry. RDB records that were never utilized are reported with 'invalid' opcodes.

Table 1950 - QUERY_RDB Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								qp_n																								08h
																																0Ch

Table 1951 - QUERY_RDB Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod		
08h	23:0	qp_n	qp number.	

Table 1952 - QUERY_RDB Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h
																								rdb_list_size								0Ch
rdb_context[0] (See Table 23, “RDB Context Layout,” on page 278)																																10h-4Ch
rdb_context[1] (See Table 23, “RDB Context Layout,” on page 278)																																50h-8Ch
...																																90h-...

Table 1953 - QUERY_RDB Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	7:0	rdb_list_size	Size of the rdb list	
10h-...	512	rdb_context[...] (See Table 23, “RDB Context Layout,” on page 278)	RDB context list. Size indicated by rdb_list_size.	

29.10.17 SEND_NVMMF_CC - Send a NVMMF Command Capsule via NVME SQ

Note: This section is preliminary and subject to change.

This command allows software to send a NVMMf command capsule and receive a response capsule on a QP that is created on behalf of a NVME function. Note that the QP must be in RTS state when submitting the CC.

In order to get the CC response, QUERY_NVME_CC_RESPONSE command must be run by software. See Section 29.10.18, “QUERY_NVMMF_CC_RESPONSE - Query NVMMF Command Capsule Response,” on page 1938.

Note: This command is only supported when `HCA_CAP.Device_Emulation_CAP.nvme_offload_type_command_capsule==1`.

Table 1954 - SEND_NVMMF_CC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhea_tunnel_id																op_mod																04h
nvme_sq																																08h
																																0Ch
																ext_data_length																10h
ext_data_umem_id																																14h
ext_data_umem_offset																																18h-1Ch
																																20h-3Ch
nvmf_cc																																40h-7Ch

Table 1955 - SEND_NVMMF_CC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	nvme_sq	The NVME SQ to send the message on behalf.	
10h	15:0	ext_data_length	Length of ext data. The data is allocated in ext_data_umem_id/ext_data_umem_offset. Device copies this data and update the SGL inside the CC to point to it.	
14h	31:0	ext_data_umem_id	UMEM ID of the ext data. Valid only when ext_data_length>0	
18h-1Ch	64	ext_data_umem_offset	Indicates offset in allocated umem id. Valid only when ext_data_length>0.	

Table 1955 - SEND_NVMMF_CC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
40h-7Ch	512	nvmf_cc	The actual command capsule.	

Table 1956 - SEND_NVMMF_CC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1957 - SEND_NVMMF_CC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.10.18 QUERY_NVMMF_CC_RESPONSE - Query NVMMF Command Capsule Response

Note: This section is preliminary and subject to change.

This command allows software to query a NVMMf command capsule response of a sent CC.

This command is only supported when HCA_CAP.Device_Emulation_CAP.nvme_offload_type_command_capsule==1. See Table 1607, “NVME Device Emulation Capabilities Layout,” on page 1787.

Table 1958 - QUERY_NVMMF_CC_RESPONSE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
nvme_sq																																08h
																																0Ch

Table 1959 - QUERY_NVME_CC_RESPONSE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	nvme_sq	The NVME SQ.	

Table 1960 - QUERY_NVME_CC_RESPONSE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																												cc_respon- se_status				0Ch
cc_response																																10h-1Ch
																																20h-3Ch

Table 1961 - QUERY_NVME_CC_RESPONSE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	3:0	cc_response_status	0x0: NO_CC_IN_FLIGHT 0x1: PENDING - still pending response. 0x2: SUCCESS 0x3: ERROR	
10h-1Ch	128	cc_response		

29.10.19 QUERY_EMULATED_FUNCTIONS_INFO - Query Emulated Functions Information

Note: This section is preliminary and subject to change.

This command allows software to query information about the host emulated functions that can be managed by the emulation manager

This command is only supported when `HCA_CAP.X_device_emulation_manager==1` where X is the device type. See Table 1571, “HCA Capabilities Layout,” on page 1671.

Table 1962 - QUERY_EMULATED_FUNCTIONS_INFO Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 1963 - QUERY_EMULATED_FUNCTIONS_INFO Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Indicates the emulated device type 0x0: NVME_DEVICE 0x1: VIRTIO_NET_DEVICE 0x2: VIRTIO_BLK_DEVICE	

Table 1964 - QUERY_EMULATED_FUNCTIONS_INFO Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																								num_emulated_pfs								0Ch

Table 1964 - QUERY_EMULATED_FUNCTIONS_INFO Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
emulated_pf_info[0] (See Table 1966, “EMULATED_PF_INFO Structure Layout,” on page 1941)																																10h-14h
emulated_pf_info[1] (See Table 1966, “EMULATED_PF_INFO Structure Layout,” on page 1941)																																18h-1Ch
...																																20h-...

Table 1965 - QUERY_EMULATED_FUNCTIONS_INFO Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	7:0	num_emulated_pfs	Number of Emulated Physical Functions.	
10h-...	64	emulated_pf_info[..](See Table 1966, “EMULATED_PF_INFO Structure Layout,” on page 1941)	List of Emulated PFs (and VFs if they exist) information. Number of entries in this list is reported in num_emulated_pfs. See See Table 1966, “EMULATED_PF_INFO Structure Layout,” on page 1941	

Table 1966 - EMULATED_PF_INFO Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pf_pci_number																pf_vhca_id																00h
num_of_vfs																vfs_base_vhca_id																04h

Table 1967 - EMULATED_PF_INFO Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	pf_pci_number	PCI physical function number	
	15:0	pf_vhca_id	VHCA Identifier of nvme PF	
04h	31:16	num_of_vfs	Number of emulated VFs belong to pf_pci_number.	
	15:0	vfs_base_vhca_id	VHCA Identifier of the first emulated VF.	

29.11 REGEXP Commands

The following commands are used for controlling REGEXP engines. For more details see TBD.
All commands are supported when `HCA_CAP.regexp==1`.

Table 1968 - REGEXP Commands Overview

Mnemonic	Command	Description	Reference
SET_REGEX-P_PARAMS	SET REGEXP PARAMS	SET REGEXP PARAMETERS	29.11.1 SET_REGEX_PARAMS – Set REGEXP PARAMS
QUERY_REGEX-P_PARAMS	QUERY REGEXP PARAMS	QUERY REGEXP PARAMETERS	29.11.2 QUERY_REGEX_PARAMS – QUERY REGEXP PARAMS
SET_REGEX-P_REGISTER	SET REGEXP REGISTER	WRITE REGEXP REGISTER	29.11.3 SET_REGEX_REGISTER – Set REGEXP Register
QUERY_REGEX-P_REGISTER	QUERY REGEXP REGISTER	QUERY REGEXP REGISTER	29.11.4 QUERY_REGEX_REGISTER – QUERY REGEXP Register

29.11.1 SET_REGEX_PARAMS – Set REGEXP PARAMS

Table 1969 - SET_REGEX_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																								engine_id				08h				
field_select																														0Ch		
regexp_params(See Table 1973, “REGEXP_PARAMS Layout,” on page 1944)																														10h-3Ch		

Table 1970 - SET_REGEX_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	7:0	engine_id	Engine Identifier. Must be less than HCA_CAP.regex_num_of_engines.
0Ch	31:0	field_select	Bitmask Indicates which fields to modify in regex_params context Bit 0: db_umem - db_umem_free, db_umem_id and db_umem_offset Bit 1: stop_engine
10h-3Ch	384	regex_params(See Table 1973, “REGEX-P_PARAMS Layout,” on page 1944)	REGEXP Parameters Context

Table 1971 - SET_REGEX_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1972 - SET_REGEX_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

Table 1973 - REGEXP_PARAMS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
db_umem_free																															stop_engine	00h
	db_umem_id																														04h	
	db_umem_offset																														08h-0Ch	
																															10h-2Ch	

Table 1974 - REGEXP_PARAMS Field Descriptions

Offset	Bits	Name	Description
00h	31	db_umem_free	If set, indicates to disconnect the specific db_umem_id from regexp database.
	0	stop_engine	When set to 1, device will stop engine and return command after on the fly works are done. When set to 0, device will return engine to work.
04h	31:0	db_umem_id	REGEXP DataBase umem_id
08h-0Ch	64	db_umem_offset	REGEXP DataBase umem_offset

29.11.2 QUERY_REGEXP_PARAMS – QUERY REGEXP PARAMS

Table 1975 - QUERY_REGEXP_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								engine_id								08h

Table 1975 - QUERY_REGEXPPARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch

Table 1976 - QUERY_REGEXPPARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	7:0	engine_id	Engine Identifier. Must be less than HCA_CAP.regexpparams_num_of_engines.

Table 1977 - QUERY_REGEXPPARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																												04h				
																												08h-0Ch				
regex_params (See Table 1973, “REGEXP_PARAMS Layout,” on page 1944)																												10h-3Ch				

Table 1978 - QUERY_REGEXPPARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
10h-3Ch	384	regexpparams (See Table 1973, “REGEXP_PARAMS Layout,” on page 1944)	REGEXP Parameters Context

29.11.3 SET_REGEX_REGISTER – Set REGEXP Register

Table 1979 - SET_REGEX_REGISTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								engine_id				08h				
register_address																																0Ch
register_data																																10h
																																14h-1Ch

Table 1980 - SET_REGEX_REGISTER Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	7:0	engine_id	Engine Identifier. Must be less than HCA_CAP.regex_num_of_engines.
0Ch	31:0	register_address	Register address. Must be aligned to 4 Bytes
10h	31:0	register_data	Data to be written to register_address

Table 1981 - SET_REGEX_REGISTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 1982 - SET_REGEX_REGISTER Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.11.4 QUERY_REGEX_REGISTER – QUERY REGEXP Register

Table 1983 - QUERY_REGEX_REGISTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																							engine_id					08h				
register_address																																0Ch

Table 1984 - QUERY_REGEX_REGISTER Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	7:0	engine_id	Engine Identifier. Must be less than HCA_CAP.regex_num_of_engines.
0Ch	31:0	register_address	Register address. Must be aligned to 4 Bytes

Table 1985 - QUERY_REGEX_REGISTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 1985 - QUERY_REGEXP_REGISTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h
register_data																																0Ch

Table 1986 - QUERY_REGEXP_REGISTER Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
0Ch	31:0	register_data	Data that read from register_address

29.12 TIR Commands

The following commands are used for controlling TIR resources (TIR is discussed in detail in Section 8.12, “Transport Interface Receive (TIR),” on page 373).

Table 1987 - TIR Commands Overview

Mnemonic	Command	Description	Reference
CREATE_TIR	Create TIR	Allocate a new Transport Interface Receive context	29.12.1 CREATE_TIR – Create TIR
MODIFY_TIR	Modify TIR	Modify a Transport Interface Receive context	29.12.2 MODIFY_TIR – Modify TIR
DESTROY_TIR	Destroy TIR	Destroy a Transport Interface Receive context	29.12.3 DESTROY_TIR – Destroy TIR
QUERY_TIR	Query TIR	Retrieves a snapshot of the current Transport Interface Receive context	29.12.4 QUERY_TIR – Query TIR

29.12.1 CREATE_TIR – Create TIR

CREATE_TIR command allocates a new TIR Context. On successful completion, a *tir_number* is returned for future reference to the created TIR Context and the context is placed at the RDY state.

Table 1988 - CREATE_TIR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-1Ch
tir_context (See Table 116, “TIR Context Format,” on page 374)																																20h-10C

Table 1989 - CREATE_TIR Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
20h-10C	1920	tir_context (See Table 116, “TIR Context Format,” on page 374)	TIR Context. Required fields must be specified.

Table 1990 - CREATE_TIR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status								icm_address[63:40]																									00h
syndrome																																04h	
icm_address[39:32]								tirn																								08h	
icm_address[31:0]																																0Ch	

Table 1991 - CREATE_TIR Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
	23:0	icm_address[63:40]	ICM address[63:40] representing this TIR as a destination in Steering. Every Flow stating this TIR as a destination should point to this address. Valid only if sw_owner capability is set for NIC Rx Flow Table Type.
04h	31:0	syndrome	
08h	31:24	icm_address[39:32]	ICM address[39:32] representing this TIR as a destination in Steering. Every Flow stating this TIR as a destination should point to this address. Valid only if sw_owner capability is set for NIC Rx Flow Table Type.
08h	23:0	tirn	TIR Context Number
0Ch	31:0	icm_address[31:0]	ICM address[31:0] representing this TIR as a destination in Steering. Every Flow stating this TIR as a destination should point to this address. Valid only if sw_owner capability is set for NIC Rx Flow Table Type.

29.12.2 MODIFY_TIR – Modify TIR

MODIFY_TIR command modifies fields in the TIR Context. The command is executed atomically on the TIR Context. The command takes a bitmask to determine which fields to update in the TIR Context.

Table 1992 - MODIFY_TIR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								tirn																								08h
																																0Ch
modify_bitmask																																10h-14h
																																18h-1Ch
tir_context (See Table 116, “TIR Context Format,” on page 374)																																20h-10C

Table 1993 - MODIFY_TIR Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	tirn	TIR Context Number
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in the TIR Context. For more details See Table 120, “MODIFY_TIR Bitmask,” on page 380
20h-10C	1920	tir_context (See Table 116, “TIR Context Format,” on page 374)	TIR Context. Only fields requested by the mask should be valid. Other fields are reserved. Indirection table entries should only be provided if the relevant bitmask bit is set.

Table 1994 - MODIFY_TIR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 1995 - MODIFY_TIR Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.12.3 DESTROY_TIR – Destroy TIR

The DESTROY_TIR command destroys a TIR context and releases the related HW resources. When used with LRO, the command also terminates outstanding LRO sessions on the TIR and generates completion for them.

Table 1996 - DESTROY_TIR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
									tirn																			08h				
																																0Ch

Table 1997 - DESTROY_TIR Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	tirn	TIR Context number
0Ch	31:0	reserved	

Table 1998 - DESTROY_TIR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 1999 - DESTROY_TIR Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

Table 1999 - DESTROY_TIR Output Structure Field Descriptions

Offset	Bits	Name	Description
08h-0Ch	64	reserved	

29.12.4 QUERY_TIR – Query TIR

The QUERY_TIR command retrieves a snapshot of the current TIR context. The TIR state and values are not affected by the QUERY_TIR command.

Table 2000 - QUERY_TIR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								tirn																				08h				
																																0Ch

Table 2001 - QUERY_TIR input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	23:0	tirn	TIR Context number	
0Ch	31:0	reserved		

Table 2002 - QUERY_TIR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch

Table 2002 - QUERY_TIR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
tir_context (See Table 116, “TIR Context Format,” on page 374)																																20h-10Ch

Table 2003 - QUERY_TIR Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	
20h-10Ch	1920	tir_context (See Table 116, “TIR Context Format,” on page 374)	TIR Context.

29.13 TIS Commands

The following commands are used for controlling TIS resources (TIS is discussed in detail in Section 8.13, “Transport Interface Send (TIS),” on page 381).

Table 2004 - TIS Commands Overview

Mnemonic	Command	Description	Reference
CREATE_TIS	Create TIS	Allocate a new Transport Interface Send context	29.13.1 CREATE_TIS – Create TIS
MODIFY_TIS	Modify TIS	Modify a Transport Interface Send context	29.13.2 MODIFY_TIS – Modify TIS
DESTROY_TIS	Destroy TIS	Destroy a Transport Interface Send context	29.13.3 DESTROY_TIS – Destroy TIS
QUERY_TIS	Query TIS	Retrieves a snapshot of the current Transport Interface Send context	29.13.4 QUERY_TIS – Query TIS

29.13.1 CREATE_TIS – Create TIS

CREATE_TIS command allocates a new TIS context. Upon successful completion, a `tis_number` is returned for future reference to the created TIS context and the context is placed at RDY state.

Table 2005 - CREATE_TIS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-1Ch
tis_context (See Table 121, “TIS Context Format,” on page 382)																																20h-...

Table 2006 - CREATE_TIS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
20h-...	1280	tis_context (See Table 121, “TIS Context Format,” on page 382)	TIS Context. Required fields must be specified.

Table 2007 - CREATE_TIS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								tish																								08h
																																0Ch

Table 2008 - CREATE_TIS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	tisn	TIS Context Number
0Ch	31:0	reserved	

29.13.2 MODIFY_TIS – Modify TIS

MODIFY_TIS command modifies fields in the TIS context. The command is executed atomically on the TIS Context. The command takes a bitmask to determine which fields to update in the TIS Context. In order to enforce correct state transition, the command also takes the TIS current state as an input. If the current state from the input does not match the existing TIS state, then the command fails (return code BAD_RESOURCE_STATE) and the TIS remains unaffected.

Note: MODIFY_TIS is supported only when HCA_CAP.modify_tis==1.

Note: MODIFY_TIS is allowed only for TIS when all SQs connected to it are not in ready state.

Table 2009 - MODIFY_TIS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								tisn																				08h				
																																0Ch
modify_bitmask																																10h-14h
																																18h-1Ch
tis_context (See Table 121, “TIS Context Format,” on page 382)																																20h-...

Table 2010 - MODIFY_TIS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	tisn	TIS Context Number
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in the TIS Context. For more details, See Table 123, “MODIFY_TIS Bitmask,” on page 383
20h-...	1280	tis_context (See Table 121, “TIS Context Format,” on page 382)	TIS Context. Only fields requested by the mask should be valid. Other fields are reserved. Indirection table entries should only be provided if the relevant bitmask bit is set.

Table 2011 - MODIFY_TIS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2012 - MODIFY_TIS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.13.3 DESTROY_TIS – Destroy TIS

The DESTROY_TIS command destroys a TIS context and releases the related HW resources.

Table 2013 - DESTROY_TIS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								tisl																								08h
																																0Ch

Table 2014 - DESTROY_TIS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	tisl	TIS Context number
0Ch	31:0	reserved	

Table 2015 - DESTROY_TIS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2016 - DESTROY_TIS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	

29.13.4 QUERY_TIS – Query TIS

The QUERY_TIS command retrieves a snapshot of the current TIS Context. The TIS state and values are not affected by the QUERY_TIS command.

Table 2017 - QUERY_TIS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									tisen																			08h				
																																0Ch

Table 2018 - QUERY_TIS input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	23:0	tisen	TIS Context number	
0Ch	31:0	reserved		

Table 2019 - QUERY_TIS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
tis_context (See Table 121, “TIS Context Format,” on page 382)																																10h-...

Table 2020 - QUERY_TIS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	

Table 2020 - QUERY_TIS Output Structure Field Descriptions

Offset	Bits	Name	Description
04h	31:0	syndrome	
08h-0Ch	64	reserved	
10h-...	1280	tis_context (See Table 121, “TIS Context Format,” on page 382)	TIS Context.

29.14 Send Queue (SQ) Commands

The following commands are used for controlling SQ resources (SQ is discussed in detail in Section 8.16, “Send Queue (SQ),” on page 402).

Table 2021 - SQ Commands Overview

Mnemonic	Command	Description	Reference
CREATE_SQ	Create SQ	Allocate and initialize a new Send Queue context	29.14.1 CREATE_SQ – Create Send Queue
MODIFY_SQ	Modify SQ	Modify a Send Queue context	29.14.2 MODIFY_SQ – Modify Send Queue
DESTROY_SQ	Destroy SQ	Destroy a Send Queue context	29.14.3 DESTROY_SQ – Destroy a Send Queue Context
QUERY_SQ	Query SQ	Retrieves a snapshot of the current Send Queue context	29.14.4 QUERY_SQ – Query Send Queue

29.14.1 CREATE_SQ – Create Send Queue

CREATE_SQ command creates new Send Queue (SQ) Context. The command allocates an SQ Context. The command then takes the SQ Context from the input mailbox and uses it to initialize the newly created SQ object. Upon successful completion, an sq_number is returned for future reference to the created SQ Context. [\[TBD\] need to decide if we want to pass mask here](#)

Table 2022 - CREATE_SQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h

Table 2022 - CREATE_SQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch-1Ch
sq_context (See Table 138, “SQ Context Format,” on page 404)																																20h-...

Table 2023 - CREATE_SQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
20h...	1984	sq_context (See Table 138, “SQ Context Format,” on page 404)	SQ Context, The list of fields that should be specified in the SQ context are listed in Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407

Table 2024 - CREATE_SQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								sqn																								08h
																																0Ch

Table 2025 - CREATE_SQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	sqn	SQ Context Number
0Ch	31:0	reserved	

29.14.2 MODIFY_SQ – Modify Send Queue

MODIFY_SQ command modifies fields in the Send Queue (SQ) Context. The command is executed atomically on the SQ Context. The command takes a bitmask to determine which fields to update in the SQ Context. In order to enforce correct state transition, the command also takes the SQ current state as an input. If the current state from the input does not match the existing SQ state, then the command fails (return code BAD_RESOURCE_STATE) and the SQ remains unaffected.

Table 2026 - MODIFY_SQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
sq_state								sqn																				08h				
																																0Ch
modify_bitmask																																10h-14h
																																18-1Ch
sq_context (See Table 138, “SQ Context Format,” on page 404)																																20h-...

Table 2027 - MODIFY_SQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	31:28	sq_state	Current SQ state Encoding is the same as in SQ context
08h	23:0	sqn	SQ Context Number
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in the SQ Context. See Table 140, “CREATE_SQ and MODIFY_SQ Bitmask,” on page 407 for more details
20h-...	1984	sq_context (See Table 138, “SQ Context Format,” on page 404)	SQ Context. Only fields requested by the mask should be valid. Other fields are reserved.

Table 2028 - MODIFY_SQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2029 - MODIFY_SQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.14.3 DESTROY_SQ – Destroy a Send Queue Context

The DESTROY_SQ command destroys a Send Queue (SQ) Context and releases the related HW resources.

Table 2030 - DESTROY_SQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									sqn																		08h					
																																0Ch

Table 2031 - DESTROY_SQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0

Table 2031 - DESTROY_SQ Input Structure Field Descriptions

Offset	Bits	Name	Description
08h	23:0	sqn	SQ Context number
0Ch	31:0	reserved	

Table 2032 - DESTROY_SQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2033 - DESTROY_SQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	

29.14.4QUERY_SQ – Query Send Queue

The QUERY_SQ command retrieves a snapshot of the current SQ Context. The SQ state and values are not affected by the QUERY_SQ command.

Table 2034 - QUERY_SQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
								sqn																				08h				
																																0Ch

Table 2035 - QUERY_SQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	23:0	sqn	SQ Context number	
0Ch	31:0	reserved		

Table 2036 - QUERY_SQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
sq_context (See Table 138, “SQ Context Format,” on page 404)																																20h-...

Table 2037 - QUERY_SQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	
20h-...	1984	sq_context (See Table 138, “SQ Context Format,” on page 404)	SQ Context.

29.15 Receive Queue (RQ) Commands

The following commands are used for controlling RQ resources (RQ is discussed in detail in Section 8.14, “Receive Queue (RQ),” on page 383).

Table 2038 - RQ Commands Overview

Mnemonic	Command	Description	Reference
CREATE_RQ	Create RQ	Allocate and initialize a new Receive Queue context	29.15.1 CREATE_RQ – Create Receive Queue
MODIFY_RQ	Modify RQ	Modify a Receive Queue context	29.15.2 MODIFY_RQ – Modify Receive Queue
DESTROY_RQ	Destroy RQ	Destroy a Receive Queue context	29.15.3 DESTROY_RQ – Destroy a Receive Queue Context
QUERY_RQ	Query RQ	Retrieves a snapshot of the current Receive Queue context	29.15.4 QUERY_RQ – Query Receive Queue

29.15.1 CREATE_RQ – Create Receive Queue

CREATE_RQ command creates a new Receive Queue (RQ) Context. The command allocates an RQ Context. The command then takes the RQ Context from the input mailbox and uses it to initialize the newly created RQ object. Upon successful completion, an `rq_number` is returned for future reference to the created RQ Context. [\[TBD\] need to decide if we want to pass mask here](#)

Table 2039 - CREATE_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																																0Ch-1Ch
rq_context (See Table 124, “RQ Context Format,” on page 385)																																20h-...

Table 2040 - CREATE_RQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539

Table 2040 - CREATE_RQ Input Structure Field Descriptions

Offset	Bits	Name	Description
04h	15:0	op_mod	Must be 0
20h-...	1984	rq_context (See Table 124, “RQ Context Format,” on page 385)	RQ Context The list of fields that should be specified in the RQ context are listed in Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395

Table 2041 - CREATE_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								rqn																								08h
																																0Ch

Table 2042 - CREATE_RQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	rqn	RQ Context Number
0Ch	31:0	reserved	

29.15.2MODIFY_RQ – Modify Receive Queue

MODIFY_RQ command modifies fields in the Receive Queue (RQ) Context. The command is executed atomically on the RQ Context. The command takes a bitmask to determine which fields to update in the RQ Context. In order to enforce correct state transition, the command also takes the RQ current state as an input. If the current state from the input does not match the existing RQ state, then the command fails (return code BAD_RESOURCE_STATE) and the RQ remains unaffected.

Table 2043 - MODIFY_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				

Table 2043 - MODIFY_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rq_state								rqn																							08h	
																																0Ch
modify_bitmask																																10h-14h
																																18h-1Ch
rq_context (See Table 124, “RQ Context Format,” on page 385)																																20h-...

Table 2044 - MODIFY_RQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	31:28	rq_state	Current RQ state Encoding is the same as in RQ context
08h	23:0	rqn	RQ Context Number
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in the RQ Context. See Table 128, “CREATE_RQ and MODIFY_RQ Bitmask,” on page 395 for more details
20h-...	1984	rq_context (See Table 124, “RQ Context Format,” on page 385)	RQ Context. Only fields requested by the mask should be valid Other fields are reserved.

Table 2045 - MODIFY_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h

Table 2045 - MODIFY_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch

Table 2046 - MODIFY_RQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.15.3 DESTROY_RQ – Destroy a Receive Queue Context

The DESTROY_RQ command destroys a Receive Queue (RQ) Context and releases the related HW resources.

Table 2047 - DESTROY_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									rqn																		08h					
																																0Ch

Table 2048 - DESTROY_RQ Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	rqn	RQ Context number
0Ch	31:0	reserved	

Table 2049 - DESTROY_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2050 - DESTROY_RQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	

29.15.4 QUERY_RQ – Query Receive Queue

The QUERY_RQ command retrieves a snapshot of the current RQ Context. The RQ state and values are not affected by the QUERY_RQ command.

Table 2051 - QUERY_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									rqn																		08h					
																																0Ch

Table 2052 - QUERY_RQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	

Table 2052 - QUERY_RQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	23:0	rqn	RQ Context number	
0Ch	31:0	reserved		

Table 2053 - QUERY_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-10h
rq_context (See Table 124, “RQ Context Format,” on page 385)																																20h-...

Table 2054 - QUERY_RQ Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-10h	64	reserved	
20h-...	1984	rq_context (See Table 124, “RQ Context Format,” on page 385)	RQ Context

29.15.5 SET_DELAY_DROP_PARAMS – Set Delay Drop Parameters

This command allows setting the device in the Delay Drop state. See Section 8.14.3, “Drop-Less RQ,” on page 398

Table 2055 - SET_DELAY_DROP_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h

Table 2055 - SET_DELAY_DROP_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h
																delay_drop_timeout																0Ch

Table 2056 - SET_DELAY_DROP_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
0Ch	15:0	delay_drop_timeout	When non-zero, packet drop due to lack of receive WQEs is delayed for a period of up to the given value. Value is given in units of 100 uSec. Value 0x0 means feature is disabled, ~~value of 0xffff means an indefinite delay~~ value is saturated at 100ms.

Table 2057 - SET_DELAY_DROP_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2058 - SET_DELAY_DROP_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.15.6 QUERY_DELAY_DROP_PARAMS – Query Delay Drop Parameters

This command querying the current Delay Drop state. See Section 8.14.3, “Drop-Less RQ,” on page 398

Table 2059 - QUERY_DELAY_DROP_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2060 - QUERY_DELAY_DROP_PARAMS input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	

Table 2061 - QUERY_DELAY_DROP_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08
																delay_drop_timeout																0Ch

Table 2062 - QUERY_DELAY_DROP_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
0Ch	15:0	delay_drop_timeout	Remaining time until Delay Drop timer expires. When 0x0, the device is no longer in Delay Drop state. Value is given in units of 100 uSec.

29.16 RQT Commands

The following commands are used for controlling RQT resources (RQT is discussed in detail in Section 8.15, “RQ Table (RQT),” on page 399).

Table 2063 - RQT Commands Overview

Mnemonic	Command	Description	Reference
CREATE_RQT	Create RQT	Allocate a new RQ Table	29.16.1 CREATE_RQT – Create RQT
MODIFY_RQT	Modify RQT	Modify a RQ Table	29.16.2 MODIFY_RQT – Modify RQ Table
DESTROY_RQT	Destroy RQT	Destroy a RQ Table	29.16.3 DESTROY_RQT – Destroy RQT
QUERY_RQT	Query RQT	Retrieves a snapshot of the current RQ Table	29.16.4 QUERY_RQT – Query RQT

29.16.1 CREATE_RQT – Create RQT

CREATE_RQT command allocates a new RQT context. Upon successful completion, an *rqt_number* is returned for future reference to the created RQT Context.

Table 2064 - CREATE_RQT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-1Ch
rqt_context (See Table 130, “RQT Context Format,” on page 399)																																20h-...

Table 2065 - CREATE_RQT Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0

Table 2065 - CREATE_RQT Input Structure Field Descriptions

Offset	Bits	Name	Description
20h-...	1952	rqt_context (See Table 130, “RQT Context Format,” on page 399)	RQT Context. Required fields must be specified.

Table 2066 - CREATE_RQT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								rqtn																								08h
																																0Ch

Table 2067 - CREATE_RQT Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	rqtn	RQT Context Number
0Ch	31:0	reserved	

29.16.2MODIFY_RQT – Modify RQ Table

MODIFY_RQT command modifies fields in the RQT context. The command takes a bitmask to determine which fields to update in the RQT context.

Table 2068 - MODIFY_RQT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								rqtn																								08h
																																0Ch

Table 2068 - MODIFY_RQT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
modify_bitmask																															10h-14h	
																															18h-1Ch	
rqt_context (See Table 130, “RQT Context Format,” on page 399)																															20h-...	

Table 2069 - MODIFY_RQT Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	rqt_n	RQT Number.
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in RQT Context. Bit 0: rqt_acual_size and the rq_num list.
20h-...	1952	rqt_context (See Table 130, “RQT Context Format,” on page 399)	RQT Context. Only fields requested by the mask should be valid, other fields are reserved.

Table 2070 - MODIFY_RQT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2071 - MODIFY_RQT Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.16.3 DESTROY_RQT – Destroy RQT

The DESTROY_RQT command destroys an RQT context and releases the related HW resources.

Table 2072 - DESTROY_RQT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									rqtn																			08h				
																																0Ch

Table 2073 - DESTROY_RQT Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	rqtn	RQT Context number
0Ch	31:0	reserved	

Table 2074 - DESTROY_RQT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
status																																					00h
syndrome																																004h					

Table 2074 - DESTROY_RQT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 08h-0Ch

Table 2075 - DESTROY_RQT Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	

29.16.4 QUERY_RQT – Query RQT

The QUERY_RQT command retrieves a snapshot of the current RQT Context.

Table 2076 - QUERY_RQT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																rqtn																08h
																																0Ch

Table 2077 - QUERY_RQT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	23:0	rqtn	RQT Context number	
0Ch	31:0	reserved		

Table 2078 - QUERY_RQT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
rqt_context (See Table 130, “RQT Context Format,” on page 399)																																20h-...

Table 2079 - QUERY_RQT Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	
20h-...	1952	rqt_context (See Table 130, “RQT Context Format,” on page 399)	RQT Context

29.17 Receive Memory Pool (RMP) Commands

The following commands are used for controlling RMP resources (RMP is discussed in detail in Section 8.17, “Receive Memory Pool (RMP),” on page 411).

Table 2080 - RMP Commands Overview

Mnemonic	Command	Description	Reference
CREATE_RMP	Create RMP	Allocate and initialize a new Receive Memory Pool context	29.17.1 CREATE_RMP – Create Receive Memory Pool
MODIFY_RMP	Modify RMP	Modify a Receive Memory Pool context	29.17.2 MODIFY_RMP – Modify Receive Memory Pool
DESTROY_RMP	Destroy RMP	Destroy a Receive Memory Pool context	29.17.3 DESTROY_RMP – Destroy a Receive Memory Pool Context

Table 2080 - RMP Commands Overview (Continued)

Mnemonic	Command	Description	Reference
QUERY_RMP	Query RMP	Retrieves a snapshot of the current Receive Memory Pool context	29.17.4 QUERY_RMP – Query Receive Memory Pool

29.17.1 CREATE_RMP – Create Receive Memory Pool

CREATE_RMP command creates new Receive Memory Pool (RMP) Context. The command allocates an RMP Context. The command then takes the RMP Context from the input mailbox and uses it to initialize the newly created RMP object. On successful completion, an *rmp_number* is returned for future reference to the created RMP Context. [\[TBD\] need to decide if we want to pass mask here](#)

Table 2081 - CREATE_RMP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																																0Ch
rmp_context (See Table 142, “RMP Context Format,” on page 412)																																20h-...

Table 2082 - CREATE_RMP Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
20h-...	1984	rmp_context (See Table 142, “RMP Context Format,” on page 412)	RMP Context The list of fields that should be specified in the RMP context are listed in Table 144, “CREATE_RMP and MODIFY_RMP Bit-mask,” on page 413

Table 2083 - CREATE_RMP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								rmpn																								08h
																																0Ch

Table 2084 - CREATE_RMP Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	rmpn	RMP Context Number
0Ch	31:0	reserved	

29.17.2MODIFY_RMP – Modify Receive Memory Pool

MODIFY_RMP command modifies fields in the Receive Memory Pool (RMP) Context. The command is executed atomically on the RMP Context. The command takes a bitmask to determine which fields to update in the RMP Context. In order to enforce correct state transition, the command also takes the RMP current state as an input. If the current state from the input does not match the existing RMP state, then the command fails (return code BAD_RESOURCE_STATE) and the RMP remains unaffected.

Table 2085 - MODIFY_RMP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
rmp_state								rmpn																									08h
																																0Ch	
modify_bitmask																																10h-14h	
																																18h-1Ch	

Table 2085 - MODIFY_RMP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rmp_context (See Table 142, “RMP Context Format,” on page 412)																																20h-...

Table 2086 - MODIFY_RMP Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	31:28	rmp_state	Current RMP state Encoding is the same as in RMP context
08h	23:0	rmpn	RMP Context Number
10h-14h	64	modify_bitmask	Bitmask that controls the fields to update in the RQ Context See Table 144, “CREATE_RMP and MODIFY_RMP Bitmask,” on page 413 for more details
20h-...	1984	rmp_context (See Table 142, “RMP Context Format,” on page 412)	RMP Context. Only fields requested by the mask should be valid, other fields are reserved.

Table 2087 - MODIFY_RMP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2088 - MODIFY_RMP Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	

Table 2088 - MODIFY_RMP Output Structure Field Descriptions

Offset	Bits	Name	Description
04h	31:0	syndrome	
08h	23:0	reserved	
0Ch	31:0	reserved	

29.17.3 DESTROY_RMP – Destroy a Receive Memory Pool Context

The DESTROY_RMP command destroys a Receive Memory Pool (RMP) Context and releases the related HW resources.

Table 2089 - DESTROY_RMP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								rmpn																								08h
																																0Ch

Table 2090 - DESTROY_RMP Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	Must be 0
08h	23:0	rmpn	RMP Context number
0Ch	31:0	reserved	

Table 2091 - DESTROY_RMP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2092 - DESTROY_RMP Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-0Ch	64	reserved	

29.17.4 QUERY_RMP – Query Receive Memory Pool

The QUERY_RMP command retrieves a snapshot of the current RMP Context. The RMP state and values are not affected by the QUERY_RMP command.

Table 2093 - QUERY_RMP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								rmpn																								08h
																																0Ch

Table 2094 - QUERY_RMP input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	23:0	rmpn	RMP Context number	
0Ch	31:0	reserved		

Table 2095 - QUERY_RMP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 2095 - QUERY_RMP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08-10h
rmp_context (See Table 142, “RMP Context Format,” on page 412)																																20h-...

Table 2096 - QUERY_RMP Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h-10h	64	reserved	
20h-...	1984	rmp_context (See Table 142, “RMP Context Format,” on page 412)	RMP Context.

29.18 Flow Table Commands

The commands in Table 2097 are used for configuring Flow Tables that are used in packet processing flows. The tables are built of Flows, where a packet attempts to match one by one according to Flow order. A Flow is defined by the values of specific headers. Consecutive Flows (1 or more) matching the same headers are defined as a Flow Group.[For more information, see TBD.](#)

Table 2097 - Flow Table Commands Overview

Mnemonic	Command	Description	Reference
CREATE_FLOW_TABLE	Allocate a new Flow Table		29.18.1 CREATE_FLOW_TABLE - Allocate a New Flow Table
DESTROY_FLOW_TABLE	De-allocate a Flow Table		29.18.3 DESTROY_FLOW_TABLE - De-allocate a Flow Table
QUERY_FLOW_TABLE	Query the flow table context		29.18.5 QUERY_FLOW_TABLE - Query Flow Table
SET_FLOW_TABLE_ROOT	Set Flow Table Root		29.18.4 SET_FLOW_TABLE_ROOT - Set Flow Table Root

Table 2097 - Flow Table Commands Overview

Mnemonic	Command	Description	Reference
CREATE_FLOW_GROUP	Define a new Flow Group		29.18.6 CREATE_FLOW_GROUP - Define a New Flow Group
DESTROY_FLOW_GROUP	De-allocate a Flow Group		29.18.7 DESTROY_FLOW_GROUP - De-allocate a Flow Group
QUERY_FLOW_GROUP	Query a flow group of specific table		29.18.8 QUERY_FLOW_GROUP - Query Flow Group
SET_FLOW_TABLE_ENTRY	Set a Flow Table Entry		29.18.9 SET_FLOW_TABLE_ENTRY - Set Flow Table Entry
QUERY_FLOW_TABLE_ENTRY	Query a Flow Table Entry		29.18.10 QUERY_FLOW_TABLE_ENTRY - Query Flow Table Entry
DELETE_FLOW_TABLE_ENTRY	Invalidate a Flow Table Entry		29.18.11 DELETE_FLOW_TABLE_ENTRY - Invalidate Flow Table Entry

29.18.1 CREATE_FLOW_TABLE - Allocate a New Flow Table

The command allocates a new Flow Table for one of the stages of packet processing. It returns a handle for future reference to this table.

Table 2098 - CREATE_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																op_mod																04h	
other_vport																vport_number																08h	
																																	08h-0Ch
	table_type																																10h
																																	14h
flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)																																18h-3Ch	

Table 2099 - CREATE_FLOW_TABLE - Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	0x0: regular_flow_table 0x1: lag_demux_flow_table - A vport LAG demux flow table is a special flow table designed for LAG support allowing packets coming from vports not natively associated with the LAG master to “jump back” to the NIC_RX root flow table of their source vport. Its entries are populated implicitly by the device and it is forbidden to add entries to it explicitly by SET_FLOW_TABLE_ENTRY command. For details, see Section 15, “LAG - Link Aggregation,” on page 861
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table table_type .
10h	31:24	table_type	Table’s role in packet processing 0x0: NIC_RX 0x1: NIC_TX 0x2: ESwitch_egress_ACL 0x3: ESwitch_ingress_ACL 0x4: ESwitch_FDB 0x5: Nic_Sniffer_Rx 0x6: Nic_Sniffer_Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
18h-3Ch	320	flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)	Flow Table Context.

Table 2100 - CREATE_FLOW_TABLE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status								icm_address[63:40]																								00h
syndrome																																04h
icm_address[39:32]								table_id																								08h
icm_address[31:0]																																0Ch

Table 2101 - CREATE_FLOW_TABLE - Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
	23:0	icm_address[63:40]	ICM address[63:40] representing this Flow Table as a destination in Steering. Every Flow stating this Flow Table as a destination should point to this address. Valid only if sw_owner capability is set for this Flow Table Type. For FDB tables, indicates the Rx ICM table address
04h	31:0	syndrome	
08h	31:24	icm_address[39:32]	ICM address[39:32] representing this Flow Table as a destination in Steering. Every Flow stating this Flow Table as a destination should point to this address. Valid only if sw_owner capability is set for this Flow Table Type. For FDB tables, indicates the Rx ICM table address
08h	23:0	table_id	Table handler. Unique for the <i>table_type</i> and <i>vport_number</i> .
0Ch	31:0	icm_address[31:0]	ICM address[31:0] representing this Flow Table as a destination in Steering. Every Flow stating this Flow Table as a destination should point to this address. Valid only if sw_owner capability is set for this Flow Table Type. For FDB tables, indicates the Rx ICM table address

Table 2102 - FLOW TABLE CONTEXT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset								
reformat_en				decap_en				sw_owner				termination table				table_miss_action				level				wqe_based_flow_update												log_size				00h
								table_miss_id																								04h								
								lag_master_next_table_id																								08h								
																																0Ch-14h								
sw_owner_icm_root_1																																18h-1Ch								
sw_owner_icm_root_0																																20h-24h								

Table 2103 - FLOW TABLE CONTEXT Field Descriptions

Offset	Bits	Name	Description
00h	31	reformat_en	If set, flow table supports <i>Reformat</i> action. Supported only if Flow_Table_Properties.reformat==1. See Table 1587, “Flow Table Properties Layout,” on page 1753.
	30	decap_en	If set, flow table supports DECAP actions. Supported only if Flow_Table_Properties.decap==1. See Table 1587, “Flow Table Properties Layout,” on page 1753.
	29	sw_owner	When set at Flow Table creation, indicates that the device SW owns this table. Supported only if sw_owner capability is set for this type of Flow Table. Setting this field makes log_size, table_miss_action and table_miss_id fields reserved.
	28	termination_table	When set, this table is used to attach actions to a steering destination when it is required to replicate a single packet to multiple destinations with different packet actions. This table may only include a single rule, with completely masked match criteria (match all), with no special miss action, and with a single destination. The destination must not be a Flow Table. Supported only when HCA_CAP.termination_table==1.
	27:24	table_miss_action	Table miss action. Indicates the behavior in case of table miss. 0x0: default_miss_table - go to default miss table according to table type default mentioned in Section 8.18.4, “Characteristics of Flow Table Types,” on page 428. 0x1: identified_miss_table - go to specific table identified by miss_table_id. Supported only when Flow_Table_Properties.identified_miss_table==1. 0x2: switch_domain - go to the root table of an alternative steering table domain. The alternative domain is indicated in Section 8.18.4, “Characteristics of Flow Table Types,” on page 428. Valid only if Flow_Table_Properties.table_miss_action_domain==1.
	23:16	level	Location in table chaining hierarchy.
	15	wqe_based_flow_update	When set, the Flow Table entries are modified using Flow Update WQE. Such table has the following properties: Must set sw_owner=0x0 log_size is reserved Must have a single group that must not have PRIORITY_TABLE as access_flow_lookup_mode.
	7:0	log_size	Log 2 of the table size (given in number of flows).

Table 2103 - FLOW TABLE CONTEXT Field Descriptions

Offset	Bits	Name	Description
04h	23:0	table_miss_id	Valid when table_miss_action==identified_miss_table. Identify the next table in case of miss in current table lookup. Table type of miss_table_id must be the same as table type of current table. Flow table level requirements must be met with next table requirements where the level of miss_table_id must be > current table level as explained in Section 8.18.3.3, “Flow Table Level,” on page 421.
08h	23:0	lag_master_next_table_id	In case the packet came from the vport natively associated with the LAG master function, the flow steering process will proceed to this flow table of the LAG master. Otherwise, the flow steering process will “jump back” to the NIC_RX root flow table of the packet source vport. Valid only when op_mod ==1. See Section 15, “LAG - Link Aggregation,” on page 861.
18h-1Ch	64	sw_owner_icm_root_1	ICM address indicating the root of this Flow Table. Every Flow stating this table as a destination should point to this address. Valid only for table type ESwitch_FDB Tx ICM address domain. When sw_owner is set, this field should be set by the driver. When sw_owner is cleared this field should be queried by the driver. valid only if the table type supports SW owner.
20h-24h	64	sw_owner_icm_root_0	ICM address indicating the root of this Flow Table. Every Flow stating this table as a destination should point to this address. For ESwitch_FDB indicates the Rx ICM address domain. Valid only if the table has sw_owner set

29.18.2MODIFY_FLOW_TABLE - Modify a Flow Table

The command modifies some properties of already allocated Flow Table. Supported only when Flow_Table_Properties.flow_table_modify==1. See Table 1588, “Flow Table Properties Field Descriptions,” on page 1754.

Table 2104 - MODIFY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other_vport																vport_number																08h
																modify_field_select																0Ch
table_type																											10h					

Table 2104 - MODIFY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								table_id																							14h	
flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)																												18h-3Ch				

Table 2105 - MODIFY_FLOW_TABLE - Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager=1.
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .
0Ch	15:0	modify_field_select	Bitmask indicates which fields to modify. Bit 0: table_miss_action/table_miss_id Bit 15: lag_master_next_table_id - applicable for vport LAG demux flow table (created with op_mod=1)
10h	31:24	table_type	Table’s role in packet processing 0x0: NIC RX 0x1: NIC TX 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, “CREATE_FLOW_TABLE - Allocate a New Flow Table,” on page 1986

Table 2105 - MODIFY_FLOW_TABLE - Input Structure Field Description

Offset	Bits	Name	Description
18h-3Ch	320	flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)	Flow Table Context.

Table 2106 - MODIFY_FLOW_TABLE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2107 - MODIFY_FLOW_TABLE - Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.18.3 DESTROY_FLOW_TABLE - De-allocate a Flow Table

The command de-allocates an empty Flow Table. When completed, all resources allocated for this table are free, and the device maintains the semantics of an empty table.

Table 2108 - DESTROY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other vport																vport_number																08h
																																0Ch
table_type																																10h

Table 2108 - DESTROY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								table_id																							14h	
																															18h-3Ch	

Table 2109 - DESTROY_FLOW_TABLE - Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .
10h	31:24	table_type	Table’s role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, “CREATE_FLOW_TABLE - Allocate a New Flow Table,” on page 1986

Table 2110 - DESTROY_FLOW_TABLE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				

Table 2110 - DESTROY_FLOW_TABLE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																04h
																																08h-0Ch

Table 2111 - DESTROY_FLOW_TABLE - Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.18.4SET_FLOW_TABLE_ROOT - Set Flow Table Root

This commands defines or replaces the root table for the specified Flow Table Type, with the Flow Table given by *table_id*. The existing root table will be disconnected and all its resources will remain.

Table 2112 - SET_FLOW_TABLE_ROOT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																op_mod																04h	
other vport																vport_number																08h	
																																	0Ch
table_type															table of other vport	table_vport_number																	10h
								table_id																									14h
								underlay_qpn																									18h

Table 2112 - SET_FLOW_TABLE_ROOT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
table_eswitch_owner_vhca_id_valid																table_eswitch_owner_vhca_id																1Ch
																																20h-3Ch

Table 2113 - SET_FLOW_TABLE_ROOT - Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539.
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	0x0: Set - Set or replace the root table for the specified Flow Table Type with the Flow Table given by table_id. For ULP, set/replace the root table with table_id for specified <Flow Table type, underlay QPN>. The existing root table will be disconnected and all its resources will remain. Supported only when HCA_CAP.modify_root. See Table 1587, “Flow Table Properties Layout,” on page 1753. 0x1: Reset_to_default - reset the root flow table for the specified Flow Table type. For ULP, reset the root flow table for the specified <Flow Table type, underlay QPN> Note that for Flow Table on ETH link that means to set level 0 Flow Table (if exists) as root, for ULP NIC RX/TX Flow Table and for other Flow Table types without explicitly created level 0 Flow Table that means- no root. Supported only when HCA_CAP.reset_root_to_default==1. See Table 1587, “Flow Table Properties Layout,” on page 1753.

Table 2113 - SET_FLOW_TABLE_ROOT - Input Structure Field Descriptions

Offset	Bits	Name	Description
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager=1. For <i>table_type</i> E-switch FDB, if driver sets HCA_CAP.esw_fdb_flow_table_root_per_port, then <i>other_vport</i> must also be set.
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> . For <i>table_type</i> E-switch FDB value 0xFFFF denotes the e-switch uplink
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
	16	table_of_other_vport	If set, the flow table specified by table_id resides in the vport namespace specified by the table_vport_number field. Supported only when HCA_CAP.esw_shared_ingress_acl==1 for Ingress and Egress ACL flow tables.
	15:0	table_vport_number	Specifies the vport namespace of the table_id field. Valid only when table_of_other_vport==1.
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986 valid only when opmode==0.
18h	23:0	underlay_qpn	Underlay QP number for enhanced ULP (i.e. IPoIB) offloads. This command sets root Flow Table of the QP. When SET_FLOW_TABLE_ROOT is executed multiple times with op_mode=0, with the same table_id and with different underlay_qpns, all the underlay QPs are associated with the same root Flow Table that is identified with the table_id. Required for NIC Transmit and NIC Receive flow tables for IB port. Otherwise reserved. Note: Only the QP that is configured to perform the enhanced ULP Stateless Offloads (QPC. ulp_stateless_offloads_mode = 2) can be associated with flow tables.

Table 2113 - SET_FLOW_TABLE_ROOT - Input Structure Field Descriptions

Offset	Bits	Name	Description
1Ch	31	table_eswitch_owner_vhca_id_valid	If set, table_eswitch_owner_vhca_id is valid. Supported only when HCA_CAP.root_ft_on_other_esw=1
	15:0	table_eswitch_owner_vhca_id	The vhca_id of the e-Switch owner that owns the root flow table specified by table_id Valid only if table_eswitch_owner_vhca_id_valid=1 Note that the VHCA specified by table_eswitch_owner_vhca_id and the VHCA that the command is issued on must have the same INIT_HCA.sw_owner_id value.

Table 2114 - SET_FLOW_TABLE_ROOT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				04h
																																				08h-0Ch

Table 2115 - SET_FLOW_TABLE_ROOT - Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.18.5 QUERY_FLOW_TABLE - Query Flow Table

The command returns a Flow Table context, as it was created by “CREATE_FLOW_TABLE - Allocate a New Flow Table”.

Table 2116 - QUERY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other_vport																vport_number																08h

Table 2116 - QUERY_FLOW_TABLE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																															0Ch	
table_type																															10h	
									table_id																						14h	
																															18h-3Ch	

Table 2117 - QUERY_FLOW_TABLE Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table table_type .
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by “CREATE_FLOW_TABLE - Allocate a New Flow Table”.

Table 2118 - QUERY_FLOW_TABLE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-14h
flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)																																18h-3Ch

Table 2119 - QUERY_FLOW_TABLE Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
18h-3Ch	320	flow_table_context (See Table 2102, “FLOW TABLE CONTEXT Structure Layout,” on page 1989)	Flow Table Context.

29.18.6CREATE_FLOW_GROUP - Define a New Flow Group

The command defines a new group of Flows inside a Flow Table. The Group is defined by a range of Flows inside a Flow Table, and the parameters of all Flows in the Group should match a packet. The command returns a handle for future reference to this Flow Group.

Table 2120 - CREATE_FLOW_GROUP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																op_mod																04h	
other_vport																vport_number																08h	
																																0Ch	
table_type													group_type																				10h

Table 2120 - CREATE_FLOW_GROUP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
								table_id																				14h																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
source eswitch owner vhma id valid																												18h																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						

Table 2121 - CREATE_FLOW_GROUP - Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539

Table 2121 - CREATE_FLOW_GROUP - Input Structure Field Description

Offset	Bits	Name	Description
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table table_type .
10h	31:24	table_type	Table's role in packet processing 0x0: NIC_receive 0x1: NIC_transmit 0x2: ESwitch_egress_ACL 0x3: ESwitch_ingress_ACL 0x4: ESwitch_FDB 0x5: NIC_Sniffer_Rx 0x6: NIC_Sniffer_Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
	19:16	group_type	Defines the structure of this flow Group, the way packets perform the lookup and the way flows are added to the group 0x0: TCAM_SUBTABLE - Flows stored in a TCAM-like order, lookup and insertion are performed according to priority (flow_index) 0x1: HASH_SPLIT - Flows are stored in a hash table, packet lookup is done based on the hash result and Flow insertion is done by bucket number (flow_index) other values are reserved
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986.

Table 2121 - CREATE_FLOW_GROUP - Input Structure Field Description

Offset	Bits	Name	Description
18h	31	source_eswitch_owner_vhca_id_valid	If set, source_eswitch_owner_vhca_id in MISC_PARAMETERS is valid. Valid only when ESW_CAP.merged_eswitch==1 (See Table 1593, “E-switch Capabilities Layout,” on page 1768).
	27:26	inner_ip_type	IP version of the inner IP header. Valid if any of the inner IP header fields is included in the match criteria. 0x1: IPv4 0x2: IPv6 other values are reserved
	25:24	outer_ip_type	IP version of the outer IP header. Valid if any of the outer IP header fields is included in the match criteria. 0x1: IPv4 0x2: IPv6 other values are reserved
1Ch	31:0	start_flow_index	The first Flow included in the group.
20h	13:8	log2_match_table_size	Log (base 2) of the number of Flow resources required for the table representing the group. Must not exceed
	6	reparsed	When set, packet will be reparsed after matching a Flow in the group. Required if the actions used in this group modify the packet format.
	5:0	log2_match_table_rows	Log (base 2) of the number of rows in the table representing the group. must not be greater than log2_match_table_size.
24h	31:0	end_flow_index	The last Flow included in the group.
28h	15:0	match_definer_id	Handle of the Match Definer object, Value 0x0 indicates this field is invalid. When valid match_criteria_enable must be 0x0, For group_type HASH_SPLIT, defines the fields included in the hash calculation. For other group_type this field is reserved
3C	7:0	match_criteria_enable	Bitmask representing which of the headers and parameters in match_criteria are used in defining the Flow. Unused parameters are reserved, and do not participate in matching packets to the Flow. Bit 0: outer_headers Bit 1: misc_parameters Bit 2: inner_headers Bit 3: misc_parameters_2 Bit 4: misc_parameters_3 other: reserved

Table 2121 - CREATE_FLOW_GROUP - Input Structure Field Description

Offset	Bits	Name	Description
40h-23Ch	4096	match_criteria (See Table 2122, “Flow Table Entry Match Parameters Format,” on page 2004)	The parameters defining all the flows belonging to the group. Cleared bits indicate that the appropriate bit in the Flow is not matched to the incoming packet.

Table 2122 - Flow Table Entry Match Parameters Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2												00h-3Ch
outer_headers (See Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005)																													40h-7Ch		
misc_parameters (See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009)																													80h-BCh		
inner_headers (See Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005)																													C0h-FCh		
misc_parameters_2 (See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013)																													100h-13Ch		
misc_parameters_3 (See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013)																													140h-1FCh		

Table 2123 - Flow Table Entry Match Parameter Field Description

Offset	Bits	Name	Description
00h-3Ch	512	outer_headers (See Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005)	Outer-most packet headers.

Table 2123 - Flow Table Entry Match Parameter Field Description

Offset	Bits	Name	Description
40h-7Ch	512	misc_parameters (See Table 2130, “Flow Table Entry Match Set Misc Parameters Format,” on page 2009)	Miscellaneous packet headers and Flow parameters
80h-BCh	512	inner_headers (See Table 2124, “Flow Table Entry Match Set Layer 2-4 Format,” on page 2005)	Inner headers of the encapsulated packet, for supported encapsulation protocols.
C0h-FCh	512	misc_parameters_2 (See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013)	
100h-13Ch	512	misc_parameters_3 (See Table 2132, “Flow Table Entry Match Set Misc Parameters 2 Format,” on page 2013)	

Table 2124 - Flow Table Entry Match Set Layer 2-4 Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										00h
smac[47:16]																												00h			
smac[15:0]														ethertype														04h			
dmac[47:16]																												08h			
dmac[15:0]														first_prio		first_cfi		first_vid										0Ch			
ip_protocol								ip_dscp				ip_ecn		svlan_tag		frag		ip_version				tcp_flags						10h			
tcp_sport														tcp_dport														14h			
																						ip_ttl_hoplimit						18h			
udp_sport														udp_dport														1Ch			

Table 2124 - Flow Table Entry Match Set Layer 2-4 Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											20h-2Ch
src_ipv4_or_ipv6																														30h-3Ch		
dst_ipv4_or_ipv6																																

Table 2125 - Flow Table Entry Match Set Layer 2-4 Field Description

Offset	Bits	Name	Description
00h	31:0	smac[47:16]	Source MAC address of incoming packet.
04h	31:16	smac[15:0]	
	15:0	ethertype	Incoming packet Ethertype - this is the Ethertype following the last VLAN tag of the packet.
08h	31:0	dmac[47:16]	Destination MAC address of incoming packet.
0Ch	31:16	dmac[15:0]	
	15:13	first_prio	Priority of first VLAN tag in the incoming packet. Valid only when cvlan_tag==1 or svlan_tag==1.
	12	first_cfi	CFI bit of first VLAN tag in the incoming packet. Valid only when cvlan_tag==1 or svlan_tag==1.
	11:0	first_vid	VLAN ID of first VLAN tag in the incoming packet. Valid only when cvlan_tag==1 or svlan_tag==1.

Table 2125 - Flow Table Entry Match Set Layer 2-4 Field Description

Offset	Bits	Name	Description
10h	31:24	ip_protocol	IP protocol.
	23:18	ip_dscp	Differentiated Services Code Point derived from Traffic Class/TOS field of IPv6/v4
	17:16	ip_ecn	Explicit Congestion Notification derived from Traffic Class/TOS field of IPv6/v4
	15	cvlan_tag	The first VLAN in the packet is c-vlan (0x8100). cvlan_tag and svlan_tag cannot be set together
	14	svlan_tag	The first VLAN in the packet is s-vlan (0x8a88). cvlan_tag and svlan_tag cannot be set together.
	13	frag	Packet is an IP fragment.
	12:9	ip_version	IP version.
	8:0	tcp_flags	TCP flags. Bit 0: FIN Bit 1: SYN Bit 2: RST Bit 3: PSH Bit 4: ACK Bit 5: URG Bit 6: ECE Bit 7: CWR Bit 8: NS
14h	31:16	tcp_sport	TCP source port. tcp and udp sport/dport are mutually exclusive.
	15:0	tcp_dport	TCP destination port. tcp and udp sport/dport are mutually exclusive.
18h	7:0	ip_ttl_hoplimit	TTL and hop_limit. This field should be qualified by an appropriate <i>ethertype</i> . For Ipv6 - hop_limit For Ipv4 - ttl.
1Ch	31:16	udp_sport	UDP source port. tcp and udp sport/dport are mutually exclusive.
	15:0	udp_dport	UDP destination port. tcp and udp sport/dport are mutually exclusive.
20h-2Ch	128	src_ipv4_or_ipv6	IP source address of incoming packets. This field should be qualified by an appropriate <i>ethertype</i> . For Ipv6 - See Table 2126, “IPv6 Address Layout,” on page 2008 For Ipv4 - See Table 2128, “IPv4 Address Layout,” on page 2008

Table 2125 - Flow Table Entry Match Set Layer 2-4 Field Description

Offset	Bits	Name	Description
30h-3Ch	128	dst_ipv4_or_ipv6	IP destination address of incoming packets. This field should be qualified by an appropriate <i>ethertype</i> . For Ipv6 - See Table 2126, “IPv6 Address Layout,” on page 2008 For Ipv4 - See Table 2128, “IPv4 Address Layout,” on page 2008

Table 2126 - IPv6 Address Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ipv6																																00h-0Ch

Table 2127 - IPv6 Address Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	ipv6	Ipv6 address	

Table 2128 - IPv4 Address Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-08h
ipv4																																0Ch

Table 2129 - IPv4 Address Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	ipv4	IPv4 address	

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31							
00h	source_sqn																																						
04h	source_eswitch_owner_vhca_id														source_port																								
08h	outer_second_vid														inner_second_cfi		inner_second_vid																						
0Ch															gre_protocol														outer_emd_tag		inner_second_svlan_tag		outer_second_svlan_tag		inner_second_cvlan_tag		outer_second_cvlan_tag		
10h	gre_key_h														gre_key_l																								
14h	vxlan_vni																																						
18h	geneve_vni																												geneve_oam		geneve_tlv_option_0_exist								
1Ch															outer_ipv6_flow_label																								
20h															inner_ipv6_flow_label																								
24h	geneve_opt_len														geneve_protocol_type														geneve_c		geneve_ver		inner_ipv6_extension		outer_ipv6_extension				

Table 2130 - Flow Table Entry Match Set Misc Parameters Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0												28h
									bth_dst_qp																								
inner_esp_spi																												2Ch					
outer_esp_spi																												30h					
																												34h					
outer_emd_tag_data_0									outer_emd_tag_data_1									outer_emd_tag_data_2						outer_emd_tag_data_3				38h					
outer_emd_tag_data_4									outer_emd_tag_data_5																		3Ch						

Table 2131 - Flow Table Entry Match Set Misc Parameters Fields

Offset	Bits	Name	Description
00h	31	gre_c_present	GRE Checksum Present field
	29	gre_k_present	GRE Key Present field
	28	gre_s_present	GRE Sequence Number Present field
	27:24	source_vhca_port	VHCA port number.
	23:0	source_sqn	Source SQN
04h	31:16	source_eswitch_owner_vhca_id	Vhca_id of the eswitch owner of the source_port. Valid only when source_eswitch_owner_vhca_id_valid in CREATE_FLOW_GROUP command is set. Note that the mask bits of both the source_eswitch_owner_vhca_id and the source_port must be either all set or all cleared.
	15:0	source_port	Source port. 0xffff determines wire port.

Table 2131 - Flow Table Entry Match Set Misc Parameters Fields

Offset	Bits	Name	Description
08h	31:29	outer_second_prio	Priority of second VLAN tag in the outer header of the incoming packet. Valid only when outer_second_cvlan_tag ==1 or outer_second_svlan_tag ==1.
	28	outer_second_cfi	CFI bit of first VLAN tag in the outer header of the incoming packet. Valid only when outer_second_cvlan_tag ==1 or outer_second_svlan_tag ==1.
	27:16	outer_second_vid	VLAN ID of first VLAN tag the outer header of the incoming packet. Valid only when outer_second_cvlan_tag ==1 or outer_second_svlan_tag ==1.
	15:13	inner_second_prio	Priority of second VLAN tag in the inner header of the incoming packet. Valid only when inner_second_cvlan_tag ==1 or inner_second_svlan_tag ==1.
	12	inner_second_cfi	CFI bit of first VLAN tag in the inner header of the incoming packet. Valid only when inner_second_cvlan_tag ==1 or inner_second_svlan_tag ==1.
	11:0	inner_second_vid	VLAN ID of first VLAN tag the inner header of the incoming packet. Valid only when inner_second_cvlan_tag ==1 or inner_second_svlan_tag ==1.
0Ch	31	outer_second_cvlan_tag	The second VLAN in the outer header of the packet is c-vlan (0x8100). outer_second_cvlan_tag and outer_second_svlan_tag cannot be set together.
	30	inner_second_cvlan_tag	The second vlan in the inner header of the packet is c-vlan (0x8100). inner_second_cvlan_tag and inner_second_svlan_tag cannot be set together.
	29	outer_second_svlan_tag	The second vlan in the outer header of the packet is s-vlan (0x8a88). outer_second_cvlan_tag and outer_second_svlan_tag cannot be set together.
	28	inner_second_svlan_tag	The second vlan in the inner header of the packet is s-vlan (0x8a88). inner_second_cvlan_tag and inner_second_svlan_tag cannot be set together.
	27	outer_emd_tag	When set, the packet contains the Embedded Meta Data tag
	15:0	gre_protocol	GRE Protocol (outer).
10h	31:8	gre_key_h	GRE Key[31:8] (outer).
	7:0	gre_key_l	GRE Key [7:0] (outer).
14h	31:8	vxlان_vni	VXLAN VNI (outer).

Table 2131 - Flow Table Entry Match Set Misc Parameters Fields

Offset	Bits	Name	Description
18h	31:8	geneve_vni	GENEVE VNI field (outer)
	1	geneve_tlv_option_0_exist	
	0	geneve_oam	GENEVE OAM field (outer)
1Ch	19:0	outer_ipv6_flow_label	Flow label of incoming IPv6 packet (outer).
20h	19:0	inner_ipv6_flow_label	Flow label of incoming IPv6 packet (inner).
24h	31	outer_ipv6_extension	Outer IPv6 header has extension(s)
	30	inner_ipv6_extension	Inner IPv6 header has extension(s)
	25:24	geneve_ver	GENEVE 'ver' field (outer)
	23	geneve_c	GENEVE 'C' field (outer)
	21:16	geneve_opt_len	GENEVE OptLen (outer)
	15:0	geneve_protocol_type	GENEVE protocol type (outer)
28h	23:0	bth_dst_qp	Destination QP in BTH header.
2Ch	31:0	inner_esp_spi	SPI field in ESP header (inner).
30h	31:0	outer_esp_spi	SPI field in ESP header (outer).
38h	31:24	outer_emd_tag_data_0	The 6 bytes of Embedded Meta Data tag, valid only when outer_emd_tag is set. Byte 0 is the most outer byte.
	23:16	outer_emd_tag_data_1	
	15:8	outer_emd_tag_data_2	
	7:0	outer_emd_tag_data_3	
3Ch	31:24	outer_emd_tag_data_4	
	23:16	outer_emd_tag_data_5	

Table 2132 - Flow Table Entry Match Set Misc Parameters 2 Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											00h	
outer_first_mpls_label																				outer - first_mpls_exp		outer first mpls s bos	outer_first_mpls_ttl										
inner_first_mpls_label																				inner - first_mpls_exp		inner first mpls s bos	inner_first_mpls_ttl										04h
outer_first_mpls_over_gre_label																				outer - first_mpls_over_gre_exp		outer first mpls over gre s bos	outer_first_mpls_over_gre_ttl										08h
outer_first_mpls_over_udp_label																				outer - first_mpls_over_udp_exp		outer first mpls over udp s bos	outer_first_mpls_over_udp_ttl										0Ch
metadata_reg_c_7																												10h					
metadata_reg_c_6																												14h					
metadata_reg_c_5																												18h					
metadata_reg_c_4																												1Ch					

Table 2132 - Flow Table Entry Match Set Misc Parameters 2 Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2													20h
metadata_reg_c_3																											24h					
metadata_reg_c_2																											28h					
metadata_reg_c_1																											2Ch					
metadata_reg_c_0																											30h					
metadata_reg_a																											34h					
															ipsec_syndrome					ipsec_next_header					38h-3Ch							

Table 2133 - Flow Table Entry Match Set Misc Parameters 2 Fields

Offset	Bits	Name	Description
00h	31:12	outer_first_mpls_label	First MPLS LABEL (outer)
	11:9	outer_first_mpls_exp	First MPLS EXP (outer)
	8	outer_first_mpls_s_bos	First MPLS S_BOS (outer)
	7:0	outer_first_mpls_ttl	First MPLS TTL (outer)
04h	31:12	inner_first_mpls_label	First MPLS LABEL (inner)
	11:9	inner_first_mpls_exp	First MPLS EXP (inner)
	8	inner_first_mpls_s_bos	First MPLS S_BOS (inner)
	7:0	inner_first_mpls_ttl	First MPLS TTL (inner)

Table 2133 - Flow Table Entry Match Set Misc Parameters 2 Fields

Offset	Bits	Name	Description
008h	31:12	outer_first_mpls_over_gre_label	First MPLS over GRE LABEL (outer)
	11:9	outer_first_mpls_over_gre_exp	First MPLS over GRE EXP (outer)
	8	outer_first_mpls_over_gre_s_bos	First MPLS over GRE S_BOS (outer)
	7:0	outer_first_mpls_over_gre_ttl	First MPLS over GRE TTL (outer)
00Ch	31:12	outer_first_mpls_over_udp_label	First MPLS over UDP LABEL (outer)
	11:9	outer_first_mpls_over_udp_exp	First MPLS over UDP EXP (outer)
	8	outer_first_mpls_over_udp_s_bos	First MPLS over UDP S_BOS (outer)
	7:0	outer_first_mpls_over_udp_ttl	First MPLS over UDP TTL (outer)
10h	31:0	metadata_reg_c_7	Metadata attached to the packet
14h	31:0	metadata_reg_c_6	Metadata attached to the packet
18h	31:0	metadata_reg_c_5	Metadata attached to the packet
1Ch	31:0	metadata_reg_c_4	Metadata attached to the packet
20h	31:0	metadata_reg_c_3	Metadata attached to the packet
24h	31:0	metadata_reg_c_2	Metadata attached to the packet
28h	31:0	metadata_reg_c_1	Metadata attached to the packet
2Ch	31:0	metadata_reg_c_0	Metadata attached to the packet
30h	31:0	metadata_reg_a	Metadata reg a field. Matched with flow_table_metadata in WQE Eth segment. Number of valid bits indicated in metadata_reg_a_width Flow Table Properties, See Table 1587, “Flow Table Properties Layout,” on page 1753
34h	15:8	ipsec_syndrome	Metadata of the packet created after passing through IPsec decryption
	7:0	ipsec_next_header	Next Header field extracted from AH/ESP, valid only after passing through IPsec decryption

Table 2134 - Flow Table Entry Match Set Misc Parameters 3 Format

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										00h
inner_tcp_seq_num																												00h			
outer_tcp_seq_num																												04h			
inner_tcp_ack_num																												08h			
outer_tcp_ack_num																												0Ch			
								outer_vxlan_gpe_vni																				10h			
outer_vxlan_gpe_next_protocol								outer_vxlan_gpe_flags																				14h			
icmp_header_data																												18h			
icmpv6_header_data																												1Ch			
icmp_type								icmp_code								icmpv6_type								icmpv6_code				20h			
geneve_tlv_option_0_data																												24h			
gtpu_teid																												28h			
gtpu_msg_type								gtpu_msg_flags																				2Ch			
																												30h-3Ch			

Table 2135 - Flow Table Entry Match Set Misc Parameters 3 Fields

Offset	Bits	Name	Description
00h	31:0	inner_tcp_seq_num	
04h	31:0	outer_tcp_seq_num	
08h	31:0	inner_tcp_ack_num	
0Ch	31:0	outer_tcp_ack_num	
10h	23:0	outer_vxlan_gpe_vni	
14h	31:24	outer_vxlan_gpe_next_protocol	
	23:16	outer_vxlan_gpe_flags	

Table 2135 - Flow Table Entry Match Set Misc Parameters 3 Fields

Offset	Bits	Name	Description
18h	31:0	icmp_header_data	
1Ch	31:0	icmpv6_header_data	
20h	31:24	icmp_type	
	23:16	icmp_code	
	15:8	icmpv6_type	
	7:0	icmpv6_code	
24h	31:0	geneve_tlv_option_0_data	Geneve TLV Option data to be matched. Must be <= HCA_CAP.max_geneve_tlv_option_data_len
28h	31:0	gtpu_teid	
2Ch	31:24	gtpu_msg_type	
	23:16	gtpu_msg_flags	

Table 2136 - CREATE_FLOW_GROUP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								group_id																								08h
																																0Ch

Table 2137 - CREATE_FLOW_GROUP - Output Structure Field Description

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
08h	23:0	group_id	Group handler. Unique for the table <i>table_type</i> , <i>table_id</i> and <i>vport_number</i> .

29.18.7 DESTROY_FLOW_GROUP - De-allocate a Flow Group

The command de-allocates an empty Flow Group. When completed, all resources allocated for this group are free.

Table 2138 - DESTROY_FLOW_GROUP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																op_mod																04h	
other_vport																vport_number																08h	
																																	0Ch
	table_type																																10h
									table_id																								14h
	group_id																																18h
																																	1Ch-3C

Table 2139 - DESTROY_FLOW_GROUP Input Structure Field Description

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for e-switch egress/ingress ACL table <i>table_type</i> .

Table 2139 - DESTROY_FLOW_GROUP Input Structure Field Description

Offset	Bits	Name	Description
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986.
18h	31:0	group_id	Group handler. Created by Section 29.18.6, "CREATE_FLOW_GROUP - Define a New Flow Group," on page 2000.

Table 2140 - DESTROY_FLOW_GROUP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				004h
																																				08h-0Ch

Table 2141 - DESTROY_FLOW_GROUP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.18.8 QUERY_FLOW_GROUP - Query Flow Group

The command returns a Flow Table Group, as it was created by “CREATE_FLOW_GROUP - Define a New Flow Group”.

Table 2142 - QUERY_FLOW_GROUP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other_vport																vport_number																08h
																																0Ch
table_type																																10h
								table_id																								14h
group_id																																18h
																																1Ch-3Ch

Table 2143 - QUERY_FLOW_GROUP Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
~~04h~~ 04h	31:16	vhca_tunnel_id	When not set to 0, indicates this command is set on behalf of another vhca, and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .

Table 2143 - QUERY_FLOW_GROUP Input Structure Field Descriptions

Offset	Bits	Name	Description
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986.
18h	31:0	group_id	Group handler. Created by Section 29.18.6, "CREATE_FLOW_GROUP - Define a New Flow Group," on page 2000.

Table 2144 - QUERY_FLOW_GROUP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-14h
						inner_ip_type	outer_ip_type																									18h
start_flow_index																																1Ch
																																20h
end_flow_index																																24h
																																28h-38h
																										match_criteria_enable						3Ch
match_criteria (See Table 2122, “Flow Table Entry Match Parameters Format,” on page 2004)																																40h-23Ch

Table 2144 - QUERY_FLOW_GROUP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																240h-3FC

Table 2145 - QUERY_FLOW_GROUP Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
18h	27:26	inner_ip_type	IP version of the inner IP header. valid if any of the inner IP header fields is included in the match criteria. 0x1: IPv4 0x2: IPv6 other values are reserved
	25:24	outer_ip_type	IP version of the outer IP header. valid if any of the outer IP header fields is included in the match criteria. 0x1: IPv4 0x2: IPv6 other values are reserved
1Ch	31:0	start_flow_index	The first Flow included in the group.
24h	31:0	end_flow_index	The last Flow included in the group.
3C	7:0	match_criteria_enable	Bitmask representing which of the headers and parameters in match_criteria are used in defining the Flow. Unused parameters are reserved, and do not participate in matching packets to the Flow. 0: outer_headers 1: misc_parameters 2: inner_headers other: reserved
40h-23Ch	4096	match_criteria (See Table 2122, “Flow Table Entry Match Parameters Format,” on page 2004)	The parameters defining all the flows belonging to the group. Cleared bits indicate that the appropriate bit in the Flow is not matched to the incoming packet.

29.18.9SET FLOW TABLE ENTRY - Set Flow Table Entry

The command adds a Flow to a Flow Table, inside a Flow Group. The Flow index determines the matching-order of the Flow, relative to other Flows in the table.

Table 2146 - SET FLOW TABLE ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other vport																vport_number																08h
																																0Ch
table_type																																10h
								table_id																								14h
ignore flow level																modify_enable_mask																18h
																																1Ch
flow_index																																20h
																																24h-3Ch
flow_context																																40h-...

(See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025)

Table 2147 - SET_FLOW_TABLE_ENTRY - Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539

Table 2147 - SET_FLOW_TABLE_ENTRY - Input Structure Field Descriptions

Offset	Bits	Name	Description
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	0: Set- Set new Entry. 1: Modify - Modify already existing entry. Valid only when HCA_ - CAP.modify_flow_en==1 for this table type. See Section 8.18.3.10, "Redefining a Flow," on page 426.
08h	31	other_vport	0 - access my vport - vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986.
18h	31	ignore_flow_level	When set, the Flow may point to Flow Table with any Level
	~~7~~ 15:0	modify_enable_mask	Bit Mask indicates which fields for already existing fields to modify Valid only when op_mod==modify Bit 0: action Bit 1: flow_tag Bit 2: Destination_List - (Size and List) Bit 3: Flow_Counters - (Size and List) Bit 4: packet_reformat_id Bit 5: modify_header_id Bit 6: push_vlan_tag Bit 7: push_vlan_2_tag Bit 8: ipsec_obj_id See Section 8.18.3.10, "Redefining a Flow," on page 426.
20h	31:0	flow_index	Flow index in the Flow Table. Index 0x0 is matched first.

Table 2147 - SET FLOW TABLE ENTRY - Input Structure Field Descriptions

Offset	Bits	Name	Description
40h-...	6208	flow_context (See Table 2148, “FLOW_CONTEXT Struc- ture Layout,” on page 2025)	Flow Table Entry Context

Table 2148 - FLOW_CONTEXT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
push_vlan_tag (See Table 2152, “VLAN Tag Structure Layout,” on page 2029)																															00h		
group_id																															04h		
								flow_tag																							08h		
																action															0Ch		
extended destination			flow_source				destination_list_size																								10h		
							flow_counter_list_size																								14h		
	packet_reformat_id																															18h	
	modify_header_id																															1Ch	
push_vlan_2_tag (See Table 2152, “VLAN Tag Structure Layout,” on page 2029)																															20h		
ipsec_obj_id																															24h		
																															28h-3Ch		
match_value (See Table 2122, “Flow Table Entry Match Parameters Format,” on page 2004)																															40h-23Ch		
																															240h-2FCh		

Table 2148 - FLOW_CONTEXT Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 300h...
basic_and_extended_destination_and_counter_list																																

Table 2149 - FLOW_CONTEXT Field Descriptions

Offset	Bits	Name	Description
00h	31:0	push_vlan_tag (See Table 2152, “VLAN Tag Structure Layout,” on page 2029)	VLAN tag to be pushed to the packet if the flow is matched and PUSH_VLAN <i>action</i> is set. Otherwise this field is reserved. Supported <i>tpid</i> values are 0x8100 (C-VLAN) and 0x8a88 (S-VLAN).
04h	31:0	group_id	Group handler. Created by Section 29.18.6, “CREATE_FLOW_GROUP - Define a New Flow Group,” on page 2000.
08h	23:0	flow_tag	Software flow identifier which is reported in CQE. flow_tag= 0x0 is invalid. Supported only when HCA_ - CAP.flow_tag==1. (will cause HW to report flow_tag of previous table) Note: For IPoIB when flow_tag is invalid in all tables, HW reports in CQE the underlay UD QPN.

Table 2149 - FLOW_CONTEXT Field Descriptions

Offset	Bits	Name	Description
0Ch	15:0	action	Bit mask indicating which actions to perform Bit 0: ALLOW - proceed to the default next step defined for this Flow Table. Bit 1: DROP - stop packet processing Bit 2: FWD_DEST - forward the packet to destination list according to destination_type. Bit 3: COUNT - count the packet in the counters in the flow counter list. Bit 4: ~~ENCAP~~REFORMAT - Reformat the packet according to the definition in the Packet Reformat Context packet_reformat_id. Supported only for flow tables with reformat_en==1. Bit 5: DECAP - decapsulate the packet. Supported only for flow tables with decap_en==1. Bit 6: MODIFY_HEADER - Modify the packet according to actions allocated in modify_header_id. Supported only for flow tables with modify_header==1. ~~Note that REFORMAT and MODIFY_HEADER cannot be set together.~~ Bit 7: POP_VLAN - pop the outer-most VLAN tag. Supported only for flow tables with pop_vlan==1. Bit 8: PUSH_VLAN - push an outer-most VLAN tag. Supported only for flow tables with push_vlan==1. Bit 9: fpga_vendor_acceleration - Send to FPGA vendor acceleration, Bit 10: POP_VLAN_2 - pop the outer-most VLAN tag. Supported only for flow tables with pop_vlan_2==1. pop_vlan_2 is illegal if not setting also pop_vlan Bit 11: PUSH_VLAN_2 - push an outer-most VLAN tag. Supported only for flow tables with push_vlan2==1. push_vlan_2 is illegal if not setting also push_vlan. push_vlan_2 always happens after push_vlan, so push_vlan_2_tag will be the outermost ~~Bit 12: IPSEC_OFFLOAD - supported when HCA_CAP.ipsec_encrypt==1 or ipsec_decrypt==1~~ Bit 12: IPSEC_DECRYPT - supported when HCA_CAP.ipsec_encrypt==1 Bit 13: IPSEC_ENCRYPT - supported when HCA_CAP.ipsec_encrypt==1

Table 2149 - FLOW_CONTEXT Field Descriptions

Offset	Bits	Name	Description
10h	31	extended_destination	Indicates if destination and counter list is basic(8 Bytes) or Extended (16 bytes). 0x0: destination_and_counter_lists 0x1: extended_destination_and_counter_lists
	29:28	flow_source	Specifies known origin of packets matching this flow. 0x0: ANY_VPORT 0x1: UPLINK 0x2: LOCAL_VPORT Other values are reserved. Valid only if HCA_CAP.flow_source is set.
	23:0	destination_list_size	Handler of the size of destination list. Must be non-zero if action [FWD_DEST] is set, otherwise reserved.
14h	23:0	flow_counter_list_size	Number of flow counters following the destination list. Must be non-zero if action [COUNT] is set, otherwise reserved.
18h	31:0	packet_reformat_id	Identifier of the Reformat Context Identifier created using ALLOC_PACKET_REFORMAT_HEADER. Valid when action REFORMAT is set.
1Ch	31:0	modify_header_id	When action includes MODIFY_HEADER: Modify header identifier. allocated using ALLOC_MODIFY_HEADER_CONTEXT command.
20h	31:0	push_vlan_2_tag (See Table 2152, “VLAN Tag Structure Layout,” on page 2029)	VLAN tag to be pushed to the packet if the flow is matched and PUSH_VLAN action is set. Otherwise this field is reserved. Supported tpid values are 0x8100 (C-VLAN) and 0x8a88 (S-VLAN). push_vlan_2 always happens after push_vlan, so push_vlan_2_tag will be the outermost.
24h	31:0	ipsec_obj_id	IPSEC object ID. Valid only when action[IPSEC_DECRYPT]==1, or action[IPSEC_ENCRYPT]==1
40h-23Ch	4096	match_value (See Table 2122, “Flow Table Entry Match Parameters Format,” on page 2004)	Values of the packet headers and parameters that should match the Flow. Valid fields are defined by the match_criteria associated with the Flow Group. Other fields are reserved.
300h-...	64	~~destination_and_counter_lists/~~ ~~extended_destination_and_counter_lists-~~ basic_and_extended_destination_and_counter_list	Basic or extended destination/flow counter list. If extended_destination ==0, then entries are basic and each one of size 8B. See Table 2154, “Destination and Counter Lists Structure Layout,” on page 2030). If extended_destination ==1, then entries are extended and each one is of size 16B. See Table 2156, “Extended Destination and Counter Lists Structure Layout,” on page 2030).

Table 2150 - SET_FLOW_TABLE_ENTRY - Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2151 - SET_FLOW_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

Table 2152 - VLAN Tag Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
tpid																pcp		dei	vid								00h					

Table 2153 - VLAN Tag Field Descriptions

Offset	Bits	Name	Description
00h	31:16	tpid	Tag Protocol Identifier
	15:13	pcp	Priority Code Point
	12	dei	Drop Eligible Indicator (formerly CFI)
	11:0	vid	VLAN Identified

Table 2154 - Destination and Counter Lists Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
destination[0]/flow_counter[0]																																00h-04h
destination[1]/flow_counter[1]																																08h-0Ch
...																																10h-...

Table 2155 - Destination and Counter Lists Field Descriptions

Offset	Bits	Name	Description
00h-...	64	destination[...]/ flow_counter[...]	Destination/Flow counter list. The first <i>destination_list_size</i> entries in the list are destinations. The following <i>flow_counter_list_size</i> entries are flow counters. See Table 2158, “Destination Entry Format Structure Layout,” on page 2031 See Table 2160, “Flow Counter List Entry Format Structure Layout,” on page 2032

Table 2156 - Extended Destination and Counter Lists Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
extended_destination[0]/extended_flow_counter[0]																																00h-0Ch
extended_destination[1]/extended_flow_counter[1]																																10h-1Ch
...																																20h-...

Table 2157 - Extended Destination and Counter Lists Field Descriptions

Offset	Bits	Name	Description
00h-...	128	extended_destination[...]/extended_flow_counter[...]	Extended Destination/Flow counter list. The first <i>destination_list_size</i> entries in the list are the extended destinations. The following <i>flow_counter_list_size</i> entries are extended flow counters. See Table 2162, “Extended Destination Format Structure Layout,” on page 2033 See Table 2164, “Extended Flow Counter List Entry Format Structure Layout,” on page 2033

Table 2158 - Destination Entry Format Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset								
destination_type								destination_id																								00h								
destination_eswitch_owner_vhca_id valid								packet reformat								destination_eswitch_owner_vhca_id																								04h

Table 2159 - Destination Entry Format Field Descriptions

Offset	Bits	Name	Description
00h	31:24	destination_type	Defines the destination type. Valid only when action [FWD_DEST] is set 0x0: VPORT 0x1: FLOW_TABLE 0x2: TIR 0x3: QP 0x4: VHCA_RX 0x5: VHCA_TX 0x6: FLOW_SAMPLER
	23:0	destination_id	When destination_type == VPORT - destination is vport number. Only bits[15:0] are valid, bits[23:16] are reserved. When destination_type == FLOW_TABLE - destination is Flow Table Id. When destination_type == TIR - destination is TIR. When destination_type == QP - destination QP Number When destination_type == VHCA_RX - destination is vhca_id. Only bits[15:0] are valid. When destination_type == VHCA_TX - destination is vhca_id. Only bits[15:0] are valid. When destination_type == FLOW_SAMPLER - destination is FLOW_SAMPLER obj_id. Only bits[23:0] are valid. [Internal] Win driver set for uplink 24 bits and not 16.
04h	31	destination_eswitch_owner_vhca_id_valid	If set, destination_eswitch_owner_vhca_id is valid. Supported only when ESW_CAP.merged_eswitch==1, and can be set only when destination_type==VPORT. See Table 1593, “E-switch Capabilities Layout,” on page 1768
	30	packet_reformat	When set, the packet would be reformatted for the specific destination (as packet_reformat_id in the Extended destination entry), and after all other actions were performed. See TBD
	15:0	destination_eswitch_owner_vhca_id	The vhca_id of the eswitch owner. Valid only when destination_eswitch_owner_vhca_id_valid==1. Note that the VHCA specified by destination_eswitch_owner_vhca_id and the VHCA the command is issued on must have the same INIT_HCA.sw_owner_id value.

Table 2160 - Flow Counter List Entry Format Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
flow_counter_id																																00h

Table 2160 - Flow Counter List Entry Format Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																04h

Table 2161 - Flow Counter List Entry Format Field Descriptions

Offset	Bits	Name	Description
00h	31:0	flow_counter_id	Flow counter handle

Table 2162 - Extended Destination Format Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
destination_entry (See Table 2158, “Destination Entry Format Structure Layout,” on page 2031)																																00h-04h
packet_reformat_id																																08h
																																0Ch

Table 2163 - Extended Destination Format Field Descriptions

Offset	Bits	Name	Description
00h-04h	64	destination_entry(See Table 2158, "Destination Entry Format Structure Layout," on page 2031)	See Table 2158, "Destination Entry Format Structure Layout," on page 2031
08h	31:0	packet_reformat_id	Same as in flow context. Valid only when <i>packet_reformat</i> field in Destination entry is set.

Table 2164 - Extended Flow Counter List Entry Format Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
flow_counter_id																																00h
																																04h-0C

Table 2165 - Extended Flow Counter List Entry Format Field Descriptions

Offset	Bits	Name	Description
00h	31:0	flow_counter_id	Flow counter handle

29.18.10 QUERY_FLOW_TABLE_ENTRY - Query Flow Table Entry

The command returns a Flow Table Entry, as it was set by “SET_FLOW_TABLE_ENTRY - Set Flow Table Entry”. The Flow Format is defined by the Flow Group the Flow belongs to. Reserved fields in the original Flow should not be queried by this command.

Table 2166 - QUERY_FLOW_TABLE_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
other_vport																vport_number																08h
table_type																																10h
									table_id																							14h
																																18h-1Ch
flow_index																																20h
																																24h-3Ch

Table 2167 - QUERY_FLOW_TABLE_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539

Table 2167 - QUERY_FLOW_TABLE_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, "CREATE_FLOW_TABLE - Allocate a New Flow Table," on page 1986.
20h	31:0	flow_index	Flow index in the Flow Table.

Table 2168 - QUERY_FLOW_TABLE_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																04h				
																																08h				
																																0Ch				
																																10h				
																																14h-3Ch				
flow_context (See Table 2148, “FLOW_CONTEXT Structure Layout,” on page 2025)																																40h-...				

Table 2169 - QUERY_FLOW_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
40h-...	6208	flow_context (See Table 2148, “FLOW_CONTEXT Struc- ture Layout,” on page 2025)	Flow Table Entry Context

29.18.11DELETE_FLOW_TABLE_ENTRY - Invalidate Flow Table Entry

The command renders a Flow Table Entry invalid. Once the command is completed, all resources allocated for the Flow Table Entry are freed, and no more packets could match this Flow.

Table 2170 - DELETE_FLOW_TABLE_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																op_mod																04h	
other_vport																vport_number																08h	
																																	0Ch
	table_type																																10h
									table_id																								14h
																																	18h-1Ch
	flow_index																																20h
																																24h-3C	

Table 2171 - DELETE_FLOW_TABLE_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.
	15:0	op_mod	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA_CAP.vport_group_manager=1
	15:0	vport_number	Number of other esw_vport on same e-switch. Valid only if other_vport =1 and for E-Switch egress/ingress ACL table <i>table_type</i> .
10h	31:24	table_type	Table's role in packet processing 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA
14h	23:0	table_id	Table handler. Created by Section 29.18.1, “CREATE_FLOW_TABLE - Allocate a New Flow Table,” on page 1986 .
20h	31:0	flow_index	Flow index in the Flow Table.

Table 2172 - DELETE_FLOW_TABLE_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0C

Table 2173 - DELETE_FLOW_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.18.12ALLOC_FLOW_COUNTER - Allocate a Flow Counter

ALLOCATE_FLOW_COUNTER command is used to allocate a new Flow counter set and return its handle. The allocated counter is initialized to 0x0.

Table 2174 - ALLOC_FLOW_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch
																								flow_counter_bulk								0Ch

Table 2175 - ALLOC_FLOW_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2175 - ALLOC_FLOW_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	7:0	flow_counter_bulk	Bit mask field indicates which bulk size to allocate. The returned flow_counter_id is the base of this size. If command fails, SW should try to allocate smaller bulk size. The supported bulk sizes reported in HCA_CAP.flow_counter_bulk_alloc. Note that flow_counter_bulk ==0, i.e. no bit is set indicates a single counter allocation. In DEALLOC, SW should only give the base flow counter of the bulk. this will deallocate all the bulk counters. Bit 0: BULK_128 - allocate 128 counters. Bit 1: BULK_256 - allocate 256 counters. Bit 2: BULK_512 - allocate 512 counters. Bit 3: BULK_1K - allocate 1K counters. Bit 4: BULK_2K - allocate 2K counters. Bit 5: BULK_4K - allocate 4K counters. Bit 6: BULK_8K - allocate 8K counters. Bit 7: BULK_16K - allocate 16K counters.	

Table 2176 - ALLOC_FLOW_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
flow_counter_id																																08h
																																0Ch

Table 2177 - ALLOC_FLOW_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	flow_counter_id	Flow counter handle.	
0Ch	31:0	reserved		

29.18.13 DEALLOC_FLOW_COUNTER - De-Allocate Flow Counter

This command is used to release the resources allocated for a flow counter. A *deallocated* must not be associated with any Flow.

Table 2178 - DEALLOC_FLOW_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
flow_counter_id																																08h
																																0Ch

Table 2179 - DEALLOC_FLOW_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	flow_counter_id	Flow counter handle.	
0Ch	31:0	reserved		

Table 2180 - DEALLOC_FLOW_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												004h				
																												08h-0Ch				

Table 2181 - DEALLOC_FLOW_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2181 - DEALLOC_FLOW_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h-0Ch	64	reserved		

29.18.14 QUERY_FLOW_COUNTER – Query Flow Counter

QUERY_FLOW_COUNTER command is used to query and reset the statistics collected by the flow counter.

Table 2182 - QUERY_FLOW_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
mkey																																0Ch
address																																10h-14h
clear	dump to memory	num_of_counters																														18h
		flow_counter_id																														1Ch

Table 2183 - QUERY_FLOW_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
0Ch	31:0	mkey	Memory key where the flow counters are dumped in case dump_to_memory==1.	
10h-14h	64	address	VA address of memory where dumping the flow counters is started in case dump_to_memory==1.	

Table 2183 - QUERY_FLOW_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
18h	31	clear	Read and reset counter value. When set to 1, HW will reset counters. Valid only when number_of_flow_counter==0.	
	30	dump_to_memory	If set, device will dump counters to the memory specified by mkey+address instead of returning them in command output mailbox. Supported only when HCA_CAP.flow_counters_dump==1.	
	29:0	num_of_counters	Number of flow counters to be read. 0 indicates reading a single flow counter. When dump_to_memory==0, the maximum value is $2^{\text{Flow\_Table\_Cap.log\_max\_flow\_counter\_bulk}}$. See Table 1587, “Flow Table Properties Layout,” on page 1753 . When dump_to_memory==1, the maximum value is HCA_CAP.max_flow_counter. Must be aligned to 4.	
1Ch	31:0	flow_counter_id	Flow counter handle (or first flow counter handle in case num_of_counters!=0). When num_of_counters!=0, must be aligned to 4.	

Table 2184 - QUERY_FLOW_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												04h				
																												08h-0Ch				
flow_statistics[0] (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																												10h-1Ch				
flow_statistics[1] (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																												20h-2Ch				
																												30h-...				

Table 2185 - QUERY_FLOW_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	128	flow_statistics[...] (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	List of flow counters. The size of the list is equal to num_of_counters. Each flow counter reports the number of packets/octets that hit the Flow Entries this Flow counter was associated with. Note that unallocated flow counters will be returned with unknown value.	

29.18.15ALLOC_PACKET_REFORMAT_CONTEXT - Allocate Packet Reformat Context

ALLOC_PACKET_REFORMAT_CONTEXT command is used to allocate a new packet reformat context. This context is used by flow table to perform packet reformatting.

Table 2186 - ALLOC_PACKET_REFORMAT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-18h
packet_reformat_context (See Table 2190, “PACKET_REFORMAT_CONTEXT Input Structure Layout,” on page 2044)																																1Ch-...

Table 2187 - ALLOC_PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2187 - ALLOC_PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
1Ch-...	31:0	packet_reformat_context[...] (See Table 2190, “PACKET_REFORMAT_CONTEXT Input Structure Layout,” on page 2044)	Packet Reformat Context to be allocated.	

Table 2188 - ALLOC_PACKET_REFORMAT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
packet_reformat_id																																08h
																																0Ch

Table 2189 - ALLOC_PACKET_REFORMAT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	packet_reformat_id	Packet Reformat Context identifier. Used in flow table with <i>Reformat</i> action.	

Table 2190 - PACKET_REFORMAT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
reformat_type																						reformat_data_size								00h		
																reformat_data[0]						reformat_data[1]						04h				
reformat_data[2]								reformat_data[3]								reformat_data[4]						reformat_data[5]						08h				
...																																...

Table 2191 - PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	reformat_type	Packet reformat operation 0x0: L2_TO_VXLAN - VXLAN Encapsulation, supported only when ESWITCH_CAP.vxlan_encap_decap==1. 0x1: L2_TO_NVGRE - NVGRE Encapsulation, supported only when ESWITCH_CAP.nvgre_encap_decap==1. 0x2: L2_TO_L2_TUNNEL - Generic L2 Encapsulation, supported only when ESWITCH_CAP.encap_general_header==1 or FLOW TABLE NIC CAP.encap_general_header==1. 0x3: L3_TUNNEL_TO_L2 - Generic L3 tunnel decapsulation and MAC insertion, supported only when <i>reformat_l3_tunnel_to_l2</i> is set for the Flow Table 0x4: L2_TO_L3_TUNNEL - Remove L2 headers and Generic L3 tunnel Encapsulation, supported only when <i>reformat_l2_to_l3_tunnel</i> is set for the Flow Table 0x5: ADD_ESP_TRANSPORT - MAC1 IP1 Data --> MAC1 IP1 ESP Data ESP-trailer. Supported only when HCA_CAP.reformat_add_esp_transport ==1. 0x6: L2_TO_L3_ESP_TUNNEL - MAC1 IP1 Data --> Encap Header IP1 Data ESP-trailer (Encap header should include ESP header). Supported only when HCA_CAP.reformat_l2_to_l3_esp_tunnel ==1. 0x7: ADD_ESP_TRANSPORT_OVER_UDP - MAC1 IP1 Data --> MAC1 IP1 UDP ESP Data ESP-trailer. Supported only when HCA_CAP.reformat_add_esp_transport_over_udp ==1. 0x8: DEL_ESP_TRANSPORT - MAC1 IP1 ESP Data ESP-trailer --> MAC1 IP1 Data. Supported only when HCA_CAP.reformat_del_esp_transport ==1. 0x9: L3_ESP_TUNNEL_TO_L2 - Encap Header ESP IP1 Data ESP-trailer --> MAC1 IP1 Data. Supported only when HCA_CAP.reformat_l3_esp_tunnel_to_l2 ==1. 0xA: DEL_ESP_TRANSPORT_OVER_UDP - MAC1 IP1 UDP ESP Data ESP-trailer --> MAC1 IP1 Data. Supported only when HCA_CAP.reformat_del_esp_transport_over_udp ==1.	

Table 2191 - PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	9:0	reformat_data_size	The amount of bytes used for reformatting the packet. Must be <=ESWITCH_CAP.max_encap_header_size. For L3_TUNNEL_TO_L2 must be 14 or 16 for MAC or MAC & VLAN respectively For ADD_ESP_TRANSPORT size should be 8B (ESP) For L2_TO_L3_ESP_TUNNEL size should be at least 42B (MAC+IP+ESP) For ADD_ESP_TRANSPORT_OVER_UDP size should be 16B (UDP+ESP) For L3_ESP_TUNNEL_TO_L2 size must be at least 14B (MAC)	
04h-...	15:8	reformat_data[...]	For L2_TO_TUNNEL_* indicates the tunnel Encapsulation header. For L3_TUNNEL_TO_L2 indicates the L2 header added to the packet after the tunnel decapsulation. For ADD_ESP_TRANSPORT should hold the ESP header. For L2_TO_L3_ESP_TUNNEL should include at least MAC, IP and ESP. For ADD_ESP_TRANSPORT_OVER_UDP should hold UDP and ESP headers. For DEL_ESP_TRANSPORT no data. For L3_ESP_TUNNEL_TO_L2 should include at least MAC header. For DEL_ESP_TRANSPORT_OVER_UDP no data. The size of this field is indicated by reformat_data_size.	

29.18.16DEALLOC_PACKET_REFORMAT_CONTEXT - De-Allocate Packet Reformat Context

This command is used to release the resources allocated for packet reformat context.

Table 2192 - DEALLOC_PACKET_REFORMAT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
packet_reformat_id																																08h
																																0Ch

Table 2193 - DEALLOC_PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	~~opcode~~ packet_reformat_id	Reformat Context identifier to be deallocated.	

Table 2194 - DEALLOC_PACKET_REFORMAT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				004h
																																				08h-0Ch

Table 2195 - DEALLOC_PACKET_REFORMAT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.18.17 QUERY_PACKET_REFORMAT_CONTEXT – Query Packet Reformat Context

This command is used to query an allocated packet reformat context.

Table 2196 - QUERY_PACKET_REFORMAT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
packet_reformat_id																																08h
																																0Ch-1Ch

Table 2197 - QUERY_PACKET_REFORMAT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	31:0	packet_reformat_id	Packet Reformat Context identifier.	

Table 2198 - QUERY_PACKET_REFORMAT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-18h
packet_reformat_context (See Table 2190, “PACKET_REFORMAT_CONTEXT Input Structure Layout,” on page 2044)																																1Ch-...

Table 2199 - QUERY_PACKET_REFORMAT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
1Ch-...	64	packet_reformat_context (See Table 2190, “PACKET_REFORMAT_CONTEXT Input Structure Layout,” on page 2044)	Packet Reformat Context	

29.18.18ALLOC_MODIFY_HEADER_CONTEXT - Allocate Modify Header Context

ALLOC_MODIFY_HEADER_CONTEXT command is used to allocate a context that specify modification actions to be performed on a packet header. This context is used by flow table to perform header modification actions. This commands is supported only when max_modify_header_actions!=0, and the maximum number of modify header contexts that can be allocated is reported by log_max_modify_header_context in Flow Table Properties, See Table 1587, “Flow Table Properties Layout,” on page 1753

Table 2200 - ALLOC_MODIFY_HEADER_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
table_type																							num_of_actions									0Ch
set_action[0]/add_action[0]/copy_action[0]																																10h-14h
set_action[1]/add_action[1]/copy_action[1]																																18h-1Ch
...																																20h-...

Table 2201 - ALLOC_MODIFY_HEADER_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	31:24	table_type	Table Type that uses this header context. Only flow tables with max_modify_header_actions!=0 in Flow Table Properties support modify header. See Table 1587, “Flow Table Properties Layout,” on page 1753 0x0: NIC receive 0x1: NIC transmit 0x2: E-Switch egress ACL 0x3: E-Switch ingress ACL 0x4: E-Switch FDB 0x5: Nic Sniffer Rx 0x6: Nic Sniffer Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA	
	7:0	num_of_actions	Number of entries in action list. i.e. number of actions to be performed. Must be <=max_modify_header_actions in Flow Table Properties. See Table 1587, “Flow Table Properties Layout,” on page 1753	

Table 2201 - ALLOC_MODIFY_HEADER_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-...	64	set_action[...]/ add_action[...]/ copy_action[...]	The actions to be performed. total number of actions specified by num_of_actions. The action type specified in the action entry by action_type: 0x0: reserved. 0x1: SET - write a specific data value a field. See Table 2204, “SET ACTION Input Structure Layout,” on page 2050 0x2: ADD - add a specific data value to a field. See Table 2206, “ADD ACTION Input Structure Layout,” on page 2053 0x3: COPY - copy a specify field to another one in a packet. See Table 2208, “COPY ACTION Input Structure Layout,” on page 2053	

Table 2202 - ALLOC_MODIFY_HEADER_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
modify_header_id																																08h
																																0Ch

Table 2203 - ALLOC_MODIFY_HEADER_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	modify_header_id	Modify header identifier. Used in flow table entry with MODIFY_HEADER actions.	

Table 2204 - SET ACTION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
action_type				field																offset								length				00h
data																																04h

Table 2205 - SET_ACTION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	action_type	Action Type 0x1: SET - write a specific data value a field. 0x2: ADD - add a specific data value to a field. 0x3: COPY - copy a specify field to another one in a packet.	

Table 2205 - SET_ACTION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	27:16	field	The field of packet to be modified. The supported fields reported in Header Modify Capabilities per Flow Table that uses the modify action. See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760. 0x1: OUT_SMAC_47_16 0x2: OUT_SMAC_15_0 0x3: OUT_ETHERTYPE 0x4: OUT_DMAC_47_16 0x5: OUT_DMAC_15_0 0x6: OUT_IP_DSCP 0x7: OUT_TCP_FLAGS 0x8: OUT_TCP_SPORT 0x9: OUT_TCP_DPORT 0xA: OUT_IPV4_TTL 0xB: OUT_UDP_SPORT 0xC: OUT_UDP_DPORT 0xD: OUT_SIPV6_127_96 0xE: OUT_SIPV6_95_64 0xF: OUT_SIPV6_63_32 0x10: OUT_SIPV6_31_0 0x11: OUT_DIPV6_127_96 0x12: OUT_DIPV6_95_64 0x13: OUT_DIPV6_63_32 0x14: OUT_DIPV6_31_0 0x15: OUT_SIPV4 0x16: OUT_DIPV4 0x17: OUT_FIRST_VID 0x31: IN_SMAC_47_16 0x32: IN_SMAC_15_0 0x33: IN_ETHERTYPE 0x34: IN_DMAC_47_16 0x35: IN_DMAC_15_0 0x36: IN_IP_DSCP 0x37: IN_TCP_FLAGS 0x38: IN_TCP_SPORT 0x39: IN_TCP_DPORT 0x3A: IN_IPV4_TTL 0x3B: IN_UDP_SPORT 0x3C: IN_UDP_DPORT 0x3D: IN_SIPV6_127_96 0x3E: IN_SIPV6_95_64 0x3F: IN_SIPV6_63_32 0x40: IN_SIPV6_31_0 0x41: IN_DIPV6_127_96 0x42: IN_DIPV6_95_64 0x43: IN_DIPV6_63_32 0x44: IN_DIPV6_31_0 0x45: IN_SIPV4 0x46: IN_DIPV4 0x47: OUT_IPV6_HOPLIMIT 0x48: IN_IPV6_HOPLIMIT 0x49: METADATA_REG_A 0x50: METADATA_REG_B 0x51: METADATA_REG_C_0 0x52: METADATA_REG_C_1 0x53: METADATA_REG_C_2 0x54: METADATA_REG_C_3 0x55: METADATA_REG_C_4 0x56: METADATA_REG_C_5 0x57: METADATA_REG_C_6 0x58: METADATA_REG_C_7 0x59: OUT_TCP_SEQ_NUM 0x5A: IN_TCP_SEQ_NUM 0x5B: OUT_TCP_ACK_NUM 0x5C: IN_TCP_ACK_NUM 0x5D: IPSEC_SYNDROME 0x60: LRH_SLID 0x61: LRH_DLID 0x62: GRH_FL 0x63: GRH_TClass 0x64: GRH_SGID_127_96 0x65: GRH_SGID_95_64 0x66: GRH_SGID_63_32 0x67: GRH_SGID_31_0 0x68: GRH_DGID_127_96 0x69: GRH_DGID_95_64 0x6A: GRH_DGID_63_32 0x6B: GRH_DGID_31_0 0x6C: BTH_PKey	

Table 2205 - SET_ACTION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	12:8	offset	The start offset in the field to be modified. for example for field DMAC_47_32, offset ==0 mean bit 32 of the destination mac address.	
	4:0	length	Number of bits to be written starting from offset. 0 means length of 32 bits.	
04h	31:0	data	The data to be written on the specific field. Data must be allocated starting from bit 0.	

Table 2206 - ADD ACTION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
action_type				field																												00h
data																																04h

Table 2207 - ADD ACTION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	action_type	Action Type 0x1: SET - write a specific data value a field. 0x2: ADD - add a specific data value to a field. 0x3: COPY - copy a specify field to another one in a packet.	
	27:16	field	The field of packet to be modified. The supported fields reported in Header Modify Capabilities per Flow Table that uses the modify action. See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760. Please see field enumeration as defined for field in Set Action, see Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051	
04h	31:0	data	The data to be added to the specific field. Data must be allocated starting from bit 0.	

Table 2208 - COPY ACTION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
action_type				src_field																src_offset								length				00h

Table 2208 - COPY ACTION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 04h
dst_field																dst_offset																

Table 2209 - COPY ACTION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	action_type	Action Type 0x0: reserved. 0x1: SET - write a specific data value a field. 0x2: ADD - add a specific data value to a field. 0x3: COPY - copy a specify field to another one in a packet.	
	27:16	src_field	The source field of packet where data is copied from. The supported copy operations reported in Header Modify Capabilities per Flow Table that uses the modify action. See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760. Please see field enumeration as defined for field in Set Action, see Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051	
	12:8	src_offset	The start offset in the source field. for example, for DMAC_47_32, offset ==0 mean bit 32 of the destination mac address.	
	4:0	length	Number of bits to be copied starting from offset. 0 means length of 32 bits.	
04h	27:16	dst_field	The destination field of packet where data is copied to. The supported copy operations reported in Header Modify Capabilities per Flow Table that uses the modify action. See Table 1589, “Header Modify Capabilities Properties Layout,” on page 1760. Please see field enumeration as defined for field in Set Action, see Table 2205, “SET_ACTION Input Structure Field Descriptions,” on page 2051.	
	12:8	dst_offset	The start offset in the destination field. for example, for DMAC_47_32, offset ==0 mean bit 32 of the destination mac address.	

29.18.19DEALLOC_MODIFY_HEADER_CONTEXT - De-Allocate Modify Header Context

This command is used to release the resources allocated for an modifier header context.

Table 2210 - DEALLOC_MODIFY_HEADER_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
modify_header_id																																08h
																																0Ch

Table 2211 - DEALLOC_MODIFY_HEADER_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:0	modify_header_id	Modify header identifier to be deallocated.	

Table 2212 - DEALLOC_MODIFY_HEADER_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2213 - DEALLOC_MODIFY_HEADER_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.18.20QUERY_MODIFY_HEADER_CONTEXT – Query Modify Header Context

~~QUERY_ENCAP_HEADER~~ command is used to query an allocated encapsulation header.

Table 2214 - QUERY_MODIFY_HEADER_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
modify_header_id																																08h
																																0Ch-1Ch

Table 2215 - QUERY_MODIFY_HEADER_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	31:0	modify_header_id	Modify header identifier.	

Table 2216 - QUERY_MODIFY_HEADER_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-08h
table_type																						num_of_actions								0Ch		
set_action[0]/add_action[0]/copy_action[0]																																10h-14h
set_action[1]/add_action[1]/copy_action[1]																																18h-1Ch
...																																20h-...

Table 2217 - QUERY_MODIFY_HEADER_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:24	table_type	Table Type that uses this header context. 0x0: NIC_receive 0x1: NIC_transmit 0x2: ESwitch_egress_ACL 0x3: ESwitch_ingress_ACL 0x4: ESwitch_FDB 0x5: Nic_Sniffer_Rx 0x6: Nic_Sniffer_Tx 0x7: NIC_RX_RDMA 0x8: NIC_TX_RDMA	
	7:0	num_of_actions	Number of entries in action list. i.e. number of actions to be performed.	
10h-...	64	set_action[...]/ add_action[...]/ copy_action[...]	The actions to be performed. total number of actions specified by num_of_actions. The action type specified in the action entry by action_type: 0x0: reserved. 0x1: SET - write a specific data value a field. See Table 2204, "SET ACTION Input Structure Layout," on page 2050 0x2: ADD - add a specific data value to a field. See Table 2206, "ADD ACTION Input Structure Layout," on page 2053 0x3: COPY - copy a specify field to another one in a packet. See Table 2208, "COPY ACTION Input Structure Layout," on page 2053	

29.18.21 SYNC_STEERING – Synchronize Steering Tables

This command is used to assure synchronization of the device steering tables with updates made by SW insertion.

Table 2218 - SYNC_STEERING Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h

Table 2218 - SYNC_STEERING Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 08h-1Ch

Table 2219 - SYNC_STEERING Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	

Table 2220 - SYNC_STEERING Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h
syndrome																																	04h
																																	08h-0Ch

Table 2221 - SYNC_STEERING Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.19 L2 Table Commands

[TBD] Consider adding “rule type” to identify steering rule or something else (eg. ACL

This section describes the commands that SW should use to inform the device of its L2 addresses.

SW must use SET_L2_TABLE_ENTRY to set its L2 addresses in the stage defined in Section 7.2, “HCA Driver Start-up,” on page 244.

If SW gets new L2 addresses after the driver start-up stage, it must also set these addresses using the same command.

SW cannot reuse the same L2 Address entry before deleting it. That is, it cannot do SET twice before DELETE of the previous one.

Note that these commands are allowed only for eswitch manager (HCA_CAP.eswitch_manager ==1).

29.19.1SET_L2_TABLE_ENTRY - Set L2 Table Entry

The command adds an L2 address entry to the L2 table.

Table 2222 - SET_L2_TABLE_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-10h
								table_index																								14h
																																18h
																				vlan_valid	vlan											1Ch
mac_address (See Table 33, “MAC Address Layout,” on page 293)																																20h-24h
																																28h-3Ch

Table 2223 - SET_L2_TABLE_ENTRY - Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	
14h	23:0	table_index	Index in the L2 table. Must be less than HCA_CAP.log_max_l2_table.
1Ch	12	vlan_valid	When set to 1, vlan field is valid. When set to 0, vlan field is not valid. Only MAC is valid.
	11:0	vlan	Ethernet vlan

Table 2223 - SET_L2_TABLE_ENTRY - Input Structure Field Descriptions

Offset	Bits	Name	Description
20h-24h	64	mac_address (See Table 33, “MAC Address Layout,” on page 293)	Mac address

Table 2224 - SET_L2_TABLE_ENTRY - Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h
																																0Ch

Table 2225 - SET_L2_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.19.2 QUERY_L2_TABLE_ENTRY - Query L2 Table Entry

The command returns an L2 table entry, as it was set by “SET_L2_TABLE_ENTRY - Set L2 Table Entry”.

Table 2226 - QUERY_L2_TABLE_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch
								table_index																								14h
																																18h-3Ch

Table 2227 - QUERY_L2_TABLE_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	
14h	23:0	table_index	Index in the L2 table. Must be less than HCA_CAP.log_max_l2_table.

Table 2228 - QUERY_L2_TABLE_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-18h
																				vlan_valid	vlan											1Ch
mac_address (See Table 33, “MAC Address Layout,” on page 293)																																20h-24h
																																28h-3Ch

Table 2229 - QUERY_L2_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	
1Ch	12	vlan_valid	When set to 1, vlan field is valid. When set to 0, vlan field is not valid. Only MAC is valid.
	11:0	vlan	Ethernet vlan

Table 2229 - QUERY_L2_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
20h-24h	64	mac_address (See Table 33, “MAC Address Layout,” on page 293)	Mac address.

29.19.3DELETE_L2_TABLE_ENTRY - Invalidate Flow Table Entry

The command renders an L2 table entry invalid.

Table 2230 - DELETE_L2_TABLE_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																0Ch-10h
									table_index																14h							
																																18h-3C

Table 2231 - DELETE_L2_TABLE_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:16	opcode	
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539
04h	15:0	op_mod	
14h	23:0	table_index	Index in the L2 table. Must be less than HCA_CAP.log_max_l2_table.

Table 2232 - DELETE_L2_TABLE_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 2232 - DELETE_L2_TABLE_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0C

Table 2233 - DELETE_L2_TABLE_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	
04h	31:0	syndrome	

29.20 FPGA QP Commands

FPGA QP commands are used to create and control QPs on the FPGA side. See Section 8.25, “FPGA QP,” on page 530.

The figure below summarizes all QP state transitions by SW and HW. It is the responsibility of software to activate QP state transitions only at the appropriate states. However, hardware may transition a QP asynchronously into the ERR state. When a state transition is requested through the command interface and the QP is in the wrong state, the command will terminate with error status.

29.20.1 FPGA_CREATE_QP - Create FPGA QP

FPGA_CREATE_QP command is used to create a new QP on the FPGA side. The required fields are reported in Table 268, “FPGA_CREATE_QP and FPGA_MODIFY_QP Bitmask,” on page 534.

Table 2234 - FPGA_CREATE_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
																																	08h-0Ch
fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)																																	10h-6Ch

Table 2235 - FPGA_CREATE_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-6Ch	768	fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)	FPGA QP Context.	

Table 2236 - FPGA_CREATE_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								fpga_qpn																								08h
																																0Ch
fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)																																10h-6Ch

Table 2237 - FPGA_CREATE_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	fpga_qpn	FPGA QP Number	
10h-6Ch	768	fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)	FPGA QP Context	

29.20.2 FPGA_MODIFY_QP - Modify FPGA QP

FPGA_MODIFY_QP command is used to perform state transition and modify parameters in FPGA QP. The modifiable parameters are reported in table TBD.

Table 2238 - FPGA_MODIFY_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								fpga_qpn																								08h
field_select																																0Ch
fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)																																10h-6Ch

Table 2239 - FPGA_MODIFY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	fpga_qpn	FPGA QP Number.	
0Ch	31:0	field_select	Field select specifies the fields to be modified. The modifiable fields and there masks describes in Table 268, “FPGA_CREATE_QP and FPGA_MODIFY_QP Bit-mask,” on page 534.	
10h-6Ch	768	fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)	FPGA QP Context.	

Table 2240 - FPGA_MODIFY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h

Table 2240 - FPGA_MODIFY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 08h-0Ch

Table 2241 - FPGA_MODIFY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.20.3 FPGA_QUERY_QP - Query FPGA QP

FPGA_QUERY_QP command is used to query FPGA QP context.

Table 2242 - FPGA_QUERY_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								fpga_qpn																								08h
																																0Ch

Table 2243 - FPGA_QUERY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	fpga_qpn	FPGA QP number	

Table 2244 - FPGA_QUERY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h

Table 2244 - FPGA_QUERY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																04h
																																08h-0Ch
fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)																																10h-6Ch

Table 2245 - FPGA_QUERY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-6Ch	768	fpga_qpc (See Table 266, “FPGA QP Context Format,” on page 531)	FPGA QP Context.	

29.20.4FPGA_DESTROY_QP - DESTROY FPGA QP

FPGA_DESTROY_QP command is used to destroy FPGA QP.

Table 2246 - FPGA_DESTROY_QP Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								fpga_qpn																								08h
																																0Ch

Table 2247 - FPGA_DESTROY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2247 - FPGA_DESTROY_QP Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	23:0	fpga_qpn	FPGA QP number	

Table 2248 - FPGA_DESTROY_QP Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2249 - FPGA_DESTROY_QP Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.20.5 FPGA_QUERY_QP_COUNTERS - Query FPGA QP Counters

FPGA_QUERY_QP_COUNTERS command is used to query FPGA QP counters.

Table 2250 - FPGA_QUERY_QP_COUNTERS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
opcode																uid																00h					
																op_mod																04h					
clear									fpga_qpn																												08h
																																0Ch					

Table 2251 - FPGA_QUERY_QP_COUNTERS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2251 - FPGA_QUERY_QP_COUNTERS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	31	clear	If set, read and reset counters.	
	23:0	fpga_qpn	FPGA QP number	

Table 2252 - FPGA_QUERY_QP_COUNTERS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
rx_ack_packets																																10h-14h
rx_send_packets																																18h-1Ch
tx_ack_packets																																20h-24h
tx_send_packets																																28h-2Ch
rx_total_drop																																30h-34h
																																38h-6Ch

Table 2253 - FPGA_QUERY_QP_COUNTERS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-14h	64	rx_ack_packets	Total amount of received AETH packets to this QP.	

Table 2253 - FPGA_QUERY_QP_COUNTERS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
18h-1Ch	64	rx_send_packets	Total amount of received SEND packets to this QP.	
20h-24h	64	tx_ack_packets	Total amount of transmitted AETH packets from this QP.	
28h-2Ch	64	tx_send_packets	Total amount of transmitted SEND packets from this QP.	
30h-34h	64	rx_total_drop	Total amount of dropped received packets to this QP. Drop can be performed due to: TBD- add list	

29.21 AFU Command - Acceleration Functions Units commands

These commands are used to create, destroy and query AFUs. see section TBD (Haggai prepare a section).

These command are supported only when HCA_CAP.afu_owner==1.

TBD - for now the API for VFs to know which AFUs supported not defined. will assume only a single AFU with id 0 is created.

29.21.1 CREATE_AFU Command

Table 2254 - CREATE_AFU Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
protocol_id																																08h
																																0Ch
afu_params (See Table 2258, “AFU_PARAMETERS_Structure Layout,” on page 2072)																																10h-3Ch
protocol_fpga_innova2 (See Table 2260, “PROTOCOL_FPGA_INNOVA2 Structure Layout,” on page 2072)																																40h-6Ch
																																70h-FCh

Table 2255 - CREATE_AFU Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:24	protocol_id	Protocol identifier. 0x0: protocol_fpga_innova2 - See	
10h-3Ch	384	afu_params (See Table 2258, “AFU_PARAMETERS - Structure Layout,” on page 2072)	Acceleration Function Unit Parameters	
40h-6Ch	384	protocol_fpga_innova2 (See Table 2260, “PROTOCOL_FPGA_INNOVA2 Structure Layout,” on page 2072)	Protocol context according to protocol_id. For protocol_id==0, protocol_fpga_innova2	

Table 2256 - CREATE_AFU Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
								afu_id																								08h
																																0Ch

Table 2257 - CREATE_AFU Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	afu_id		

Table 2258 - AFU_PARAMETERS_Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								ieee_vendor_id																				00h				
product_version																product_id												04h				
																																08h-2Ch

Table 2259 - AFU_PARAMETERS Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:0	ieee_vendor_id	IEEE Vendor ID of AFU. 0x2C9: Mellanox	
04h	31:16	product_version	Sandbox version.	
	15:0	product_id	AFU product ID. definition depends on vendor_id. For Mellanox products 0x0: unknown 0x1: example 0x2: IPsec 0x3: TLS	

Table 2260 - PROTOCOL_FPGA_INNOVA2 Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
sbu_index																																00h
																																04h-2Ch

Table 2261 - PROTOCOL_FPGA_INNOVA2 Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	sbu_index	Sandbox index where the AFU is processed.	

29.21.2 QUERY_AFU Command

Table 2262 - QUERY_AFU Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								afu_id																								08h
																																0Ch

Table 2263 - QUERY_AFU Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	afu_id		

Table 2264 - QUERY_AFU Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
protocol_id																																08h
																																0Ch
afu_params (See Table 2258, “AFU_PARAMETERS_Structure Layout,” on page 2072)																																10h-3Ch
protocol_fpga_innova2 (See Table 2260, “PROTOCOL_FPGA_INNOVA2 Structure Layout,” on page 2072)																																40h-6Ch
																																70h-FCCh

Table 2265 - QUERY_AFU Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:24	protocol_id	Protocol identifier. 0x0: protocol_fpga_innova2 - See	
10h-3Ch	384	afu_params (See Table 2258, “AFU_PARAMETERS_ Structure Layout,” on page 2072)		
40h-6Ch	384	protocol_fpga_innova2 (See Table 2260, “PROTOCOL_FPGA_INNOVA2 Structure Layout,” on page 2072)	Protocol context according to protocol_id. For protocol_id==0, protocol_fpga_innova2 See Table 2260, “PROTOCOL_FPGA_INNOVA2 Structure Layout,” on page 2072	

29.21.3 DESTROY_AFU Command

Table 2266 - DESTROY_AFU Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
								afu_id																				08h				
																																0Ch

Table 2267 - DESTROY_AFU Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	afu_id		

Table 2268 - DESTROY_AFU Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2269 - DESTROY_AFU Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.22 CAPI Commands

29.22.1 CREATE_CAPI_PEC

The CREATE_PEC command generates a new CAPI process element context. The returned pasid index can be use as PASID in MKey contexts.

Table 2270 - CREATE_CAPI_PEC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch
capi_pec_context (See Table 2274, “CAPI_PEC_Context Structure Layout,” on page 2076)																																10h-3Ch

Table 2271 - CREATE_CAPI_PEC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-3Ch	384	capi_pec_context (See Table 2274, “CAPI_PEC_Context Structure Layout,” on page 2076)	CAPI PEC context	

Table 2272 - CREATE_CAPI_PEC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
								pasid																								08h
																																0Ch

Table 2273 - CREATE_CAPI_PEC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	pasid		

Table 2274 - CAPI_PEC_Context Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
state_register																																00h-04h
thread_id																																08h
process_id																																0Ch

Table 2274 - CAPI_PEC_Context Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
local_partition_id																																10h
																																14h-2Ch

Table 2275 - CAPI_PEC_Context Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	state_register		
08h	31:0	thread_id		
0Ch	31:0	process_id		
10h	31:0	local_partition_id		

29.22.2 QUERY_CAPI_PEC

Table 2276 - QUERY_CAPI_PEC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								pasid																				08h				
																																0Ch

Table 2277 - QEURY_CAPI_PEC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	pasid		

Table 2278 - QUERY_CAPI_PEC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch
capi_pec_context (See Table 2274, “CAPI_PEC_Context Structure Layout,” on page 2076)																																10h-3Ch

Table 2279 - QUERY_CAPI_PEC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-3Ch	384	capi_pec_context (See Table 2274, “CAPI_PEC_Context Structure Layout,” on page 2076)	CAPI PEC Context	

29.22.3 DESTROY_CAPI_PEC

Table 2280 - DESTROY_CAPI_PEC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								pasid																								08h
																																0Ch

Table 2281 - DESTROY_CAPI_PEC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	pasid		

Table 2282 - DESTROY_CAPI_PEC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2283 - DESTROY_CAPI_PEC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.23 General Object Commands

Note: This section is preliminary and subject to change.

The following commands allows managing objects in a generic manner, with a predefined set of operations: CREATE, QUERY, MODIFY and DISTROY.

One benefit of this generic command format is to enable kernel SW to cleanup (destroy) user application SW objects that are created directly with PRM commands (see the <cross reference to the UCTX chapter>), just by inspecting the user command headers, without knowing the command specific content.

The supported objects reported in HCA_CAP.general_obj_types (See Table 1571, “HCA Capabilities Layout,” on page 1671).

The object types enumeration that uses in the commands described in the table below:

Table 2284 - GENERAL OBJECT TYPES

Enumeration	Object Type Name	Link to Object Type Description	Link to Object
0x0006	Device_Emulation_Bar_Object	Section 8.27.3, “NVMe Device Emulation Object,” on page 541	Table 270, “NVMe Device Emulation Object Format,” on page 541
0x0007	NVME_ ~~Q~~ PSQ	Section 8.27.4, “NVME SQ - NVME Emulation SQ Object,” on page 542	Table 272, “NVME_SQ Object Format,” on page 543
0x0008	SW_ICM_Object	Section 8.27.10, “SW ICM Block Object,” on page 554	Table 284, “SW ICM Block Object Format,” on page 554
0x0009	NVME_CQ	Section 8.27.6, “NVME CQ - NVME Emulation CQ Object,” on page 547	Table 276, “NVME_CQ Object Format,” on page 547
0x000A	FLOW_METER		Table 286, “Flow Meter Object Format,” on page 556
0x000B	GENEVE_TLV_OPTION		Table 292, “GENEVE TLV OPTION Object Format,” on page 562
0x000C	~~ENCRYPTION_KEY~~ DEK		Table 294, “DEK Object Format,” on page 564
0x000D	VIRTIO_NET_Q	Section 8.27.18, “VIRTIO_NET_Q Object,” on page 571	Table 304, “VIRTIO_NET_Q Object Format,” on page 573
0x000E	VHCA_TUNNEL_OBJECT		Table 340, “VHCA_TUNNEL Object Format,” on page 605
0x000F	NVME_NAMESPACE		Table 274, “NVME_NAMESPACE Object Format,” on page 546
0x0010	NVME_CTRL_COUNTERS		Table 278, “NVME_CTRL_COUNTERS Object Format,” on page 549
0x0011	NVME_NAMESPACE_COUNTERS		Table 280, “NVME_NAMESPACE_COUNTERS Object Format,” on page 551
0x0012	NVME_SQ_COUNTERS		Table 282, “NVME_SQ_COUNTERS Object Format,” on page 552

Table 2284 - GENERAL OBJECT TYPES

Enumera tion	Object Type Name	Link to Object Type Description	Link to Object
0x0013	IPSEC_OFFLOAD		Table 342, “IPSEC_OFF- LOAD Object Format,” on page 606
0x0014	DEVICE_OBJECT		
0x0015	VIRTIO_NET_DEVICE_E- MULTION		
0x0016	VIRTIO_BLK_DEVICE_E- MULTION		
0x0017	VIRTQ_BLK_Q		
0x0018	MATCH_DEFINER		Table 347, “MATCH_DE- FINER Fields,” on page 611
0x0019	NVMF_BACKEND_- CONTROLLER		
0x001A	NVMF_NAMESPACE		
0x001B	NVMF_NAME- SPACE_COUNTER_SET		
0x001C	VIRTIO_Q_COUNTERs		
0x001D	IMPORT_KEK		Table 297, “IMPORT_KEK Object Format,” on page 567
0x001E	CREDENTIAL		
0x001F	CRYPTO_LOGIN		

29.23.1CREATE_GENERAL_OBJ

Table 2285 - CREATE_GENERAL_OBJ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_in_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)																																00h-0Ch
obj_context																																10h-....

Table 2286 - CREATE_GENERAL_OBJ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_in_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)	General Object input command header. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082	
10h-...	31:0	obj_context	Object context according to obj_type	

Table 2287 - CREATE_GENERAL_OBJ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)																																00h-0Ch

Table 2288 - CREATE_GENERAL_OBJ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)	General Object output command header. See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083	

Table 2289 - GENERAL_OBJ_IN_CMD_HDR Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
vhca_tunnel_id																obj_type																04h	
obj_id																																	08h
																																	0Ch

Table 2290 - GENERAL_OBJ_IN_CMD_HDR Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	obj_type	Object Type. See Table 2284, “GENERAL OBJECT TYPES,” on page 2080	
08h	31:0	obj_id	Object identifier. valid only in MODIFY, QUERY and DESTROY commands.	

Table 2291 - GENERAL_OBJ_OUT_CMD_HDR Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
obj_id																																08h
																																0Ch

Table 2292 - GENERAL_OBJ_OUT_CMD_HDR Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	obj_id	Object identifier of the created object. Valid only in CREATE command.	

29.23.2 MODIFY_GENERAL_OBJ – Modify General Object

Table 2293 - MODIFY_GENERAL_OBJ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_in_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)																																00h-0Ch
obj_context																																10h-...

Table 2294 - MODIFY_GENERAL_OBJ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_in_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)	General Object input command header. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082.	
10h-...	31:0	obj_context		

Table 2295 - MODIFY_GENERAL_OBJ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)																															00h-0Ch	

Table 2296 - MODIFY_GENERAL_OBJ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)	General Object output command header. See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083	

29.23.3 QUERY_GENERAL_OBJ

Table 2297 - QUERY_GENERAL_OBJ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_in_cmd_hdr(See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)																																00h-0Ch

Table 2298 - QEURY_GENERAL_OBJ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_in_cmd_hdr(See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)	General Object input command header. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082	

Table 2299 - QUERY_GENERAL_OBJ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)																																00h-0Ch
obj_context																																10h-...

Table 2300 - QUERY_GENERAL_OBJ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)	General Object output command header. See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083	
10h-...	31:0	obj_context		

29.23.4 DESTROY_GENERAL_OBJ

Table 2301 - DESTROY_GENERAL_OBJ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_out_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)																																00h-0Ch

Table 2302 - DESTROY_GENERAL_OBJ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_out_cmd_hdr (See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082)	General Object input command header. See Table 2289, “GENERAL_OBJ_IN_CMD_HDR Structure Layout,” on page 2082	

Table 2303 - DESTROY_GENERAL_OBJ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)																																00h-0Ch

Table 2304 - DESTROY_GENERAL_OBJ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	general_obj_out_cmd_hdr (See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083)	General Object output command header. See Table 2291, “GENERAL_OBJ_OUT_CMD_HDR Structure Layout,” on page 2083	

29.23.5 CREATE_UCTX

Table 2305 - CREATE_UCTX Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																																00h
																op_mod																04h
																																08h-0Ch
uctx_object (See Table 2309, “UCTX Object Format,” on page 2088)																																10h-3Ch

Table 2306 - CREATE_UCTX Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod		
10h-3Ch	384	uctx_object (See Table 2309, “UCTX Object Format,” on page 2088)	UCTX Object. See Table 2309, “UCTX Object Format,” on page 2088	

Table 2307 - CREATE_UCTX Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																uid																08h
																																0Ch

Table 2308 - CREATE_UCTX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2308 - CREATE_UCTX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	15:0	uid		

Table 2309 - UCTX Object Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									00h
cap																														04-2Ch

Table 2310 - UCTX Object Fields

Offset	Bits	Name	Description
00h	31:0	cap	Bit mask indicates which capabilities are allowed for this user. Bit 0x0: RAW_TX - When set the device will allow the user to create objects that enable raw packet transmission (where the user can control all packet headers). Specifically this means that the device allow the CREATE_TIS command for that user. Bit 0x1: INTERNAL_DEVICE_RESOURCES - when set, the device allows this user to access internal device resources directly. i.e. user will et these capabilities, HCA_CAP.log_steering_sw_icm_size, log_header_modify_sw_icm_size, sw_owner and ignore_flow_level.

29.23.6 QUERY_UCTX

Table 2311 - QUERY_UCTX Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																																00h

Table 2311 - QUERY_UCTX Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																op_mod																04h
																uid																08h
																																0Ch

Table 2312 - QUERY_UCTX Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod		
08h	15:0	uid		

Table 2313 - QUERY_UCTX Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch
uctx_object (See Table 2309, “UCTX Object Format,” on page 2088)																																10h-3Ch

Table 2314 - QUERY_UCTX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-3Ch	384	uctx_object (See Table 2309, “UCTX Object Format,” on page 2088)	UCTX Object	

29.23.7 DESTROY_UCTX

Table 2315 - DESTROY_UCTX Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																																00h
																op_mod																04h
																uid																08h
																																0Ch

Table 2316 - DESTROY_UCTX Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
04h	15:0	op_mod		
08h	15:0	uid		

Table 2317 - DESTROY_UCTX Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h
syndrome																																004h	
																																08h-0Ch	

Table 2318 - DESTROY_UCTX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.23.8 CREATE_UMEM

Table 2319 - CREATE_UMEM Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch
umem_context (See Table 2323, “UMEM Context Format,” on page 2092)																																10h-...

Table 2320 - CREATE_UMEM Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-...	320	umem_context (See Table 2323, “UMEM Context Format,” on page 2092)	UMEM Context.	

Table 2321 - CREATE_UMEM Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
								umem_id																								08h
																																0Ch

Table 2322 - CREATE_UMEM Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2322 - CREATE_UMEM Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	23:0	umem_id		

Table 2323 - UMEM Context Format

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0								00h-0Ch
																													10h
																													log_page_size
																													14h
																													18h-1Ch
																													20h-24h
																													28h-2Ch
																													30h-..

Table 2324 - UMEM Context Fields

Offset	Bits	Name	Description
10h	31	ats	If set, address translation service is enabled for this UMEM. Supported only when HCA_CAP.ats==1.
	4:0	log_page_size	Log (base 2) of page size in units of 4Kbyte.
14h	31:0	page_offset	Page offset in granularity of bytes.

Table 2324 - UMEM Context Fields

Offset	Bits	Name	Description
18h-1Ch	64	num_of_mtt	Number of MTT entries
20h-...	64	mtt[...] (See Table 365, “Memory Translation Table (MTT) Entry Layout,” on page 634)	

29.23.9QUERY_UMEM

Table 2325 - QUERY_UMEM Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								umem_id																								08h
																																0Ch

Table 2326 - QUERY_UMEM Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	umem_id		

Table 2327 - QUERY_UMEM Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h

Table 2327 - QUERY_UMEM Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																004h
																																08h-0C h
umem_context (See Table 2323, “UMEM Context Format,” on page 2092)																																10h-34h

Table 2328 - QUERY_UMEM Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-34h	320	umem_context (See Table 2323, “UMEM Context Format,” on page 2092)	UMEM Context	

29.23.10DESTROY_UMEM

Table 2329 - DESTROY_UMEM Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								umem_id																								08h
																																0Ch

Table 2330 - DESTROY_UMEM Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2330 - DESTROY_UMEM Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	23:0	umem_id		

Table 2331 - DESTROY_UMEM Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																															004h	
																															08h-0Ch	

Table 2332 - DESTROY_UMEM Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.24 Special QPs and Management Commands

The special QPs are:

- QP0
- QP1

For each virtual HCA port, HCA supports one QP of each of these QP types. To process messages destined to these QPs, software creates QPs of the appropriate transport type. If such QPs do not exist, special QP packets will not be forwarded to software.

The MAD_DEMUX command manages the MAD de-multiplexing configuration. The configuration demux MADS between SW and internal FW agent based on MAD classes, attributes and methods.

The MAD_IFC command is used for passing parameters in a MAD format to be executed by the hardware SMA and all other agents. This command should be used to retrieve or configure Portinfo, HCAinfo, etc. For instance, to read Portinfo attributes, the software should compose a GetPortInfo MAD, and use the MAD_IFC command to execute the command and retrieve the port values. The following commands are used for configuring and operating the HCA special QPs: QP0, QP1, Raw Datagram and Raw IPv6.

The following commands configure and operate the HCA special QPs.

Table 2333 - Special QPs and Management Commands

Mnemonic	Command	Description	Reference
QUERY_MAD_DEMUX	Query MAD Demux	Retrieves the MAD de-multiplexing configuration	Section 29.24.1, on page 2096
SET_MAD_DEMUX	Set MAD Demux	Sets the MAD de-multiplexing configuration	Section 29.24.2, on page 2100
MAD_IFC	Management Datagram Interface	Takes a MAD packet from the Input Mailbox, processes it, and stores the response MAD in the Output Mailbox	Section 29.24.3, on page 2101

29.24.1 QUERY_MAD_DEMUX

The QUERY_MAD_DEMUX retrieves the MAD de-multiplexing configuration. The command returns the MAD classes, attributes and methods that are forwarded to the host, MADs that are not filtered to be passed to the host are processed internally by FW.

Table 2334 - QUERY_MAD_DEMUX Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																														08h-0Ch		

Table 2335 - QUERY_MAD_DEMUX Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 2336 - QUERY_MAD_DEMUX Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch
mad_demux_parameters_block																																10h-...

Table 2337 - QUERY_MAD_DEMUX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	31:0	mad_demux-parameters_block		

Table 2338 - MAD_DEMUX Parameters Block Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	11	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0										00h
																											00h				
						demu x_mo de																filters_count	04h								
valid	filter_id												actio n					method_select	method				08h	Filter 1							
	class												attribute								0Ch										
valid	filter_id												actio n					method_select	method				10h	Filter 2							
	class												attribute								14h										

Table 2338 - MAD_DEMUX Parameters Block Layout

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	Offset
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2		0											
valid	filter_id																action							method_select	method						(8N)h	Filter N
	class																attribute												04h+(8N)h			

Table 2339 - MAD_DEMUX Parameters Block Field Descriptions

Offset	Bits	Name	Description	Access
04h	17:16	demux_mode	0 - Forward all MADs to the host. 1 - Do not forward MADs to the host, all MADs are processed internally. 2 - Forward according to the demux filters.	
	5:0	filter_count	The number of filter entrys.	

Table 2339 - MAD_DEMUX Parameters Block Field Descriptions

Offset	Bits	Name	Description	Access
(8n)h	31	valid	Indicates that the filter is valid, when a filter_id is set as not valid it will be removed from the filters table	
	29:24	filter_id	A filter_id to be associated with the filter. The filter_id is used in order to modify or invalidate a filter.	
	15:14	action	0 - Forward to the host. 1-3 reserved	
	9:8	method_select	The method_select determines the usage of the method field in the filter: 0 - Specific method 1 - Common methods filter 3 - All methods	
	7:0	method	The method field usage depends of teh method_select field: 1. When method_select = 0 - Specific method, the method field holds the method value to filter. 2. When method select = 1 - Common method filter, the method field holds a bitmask a bit per method, when set the method is filtered: 0 - Get() - 0x01 1 - Set() - 0x02 2 - GetResp() - 0x81 3 - Send() - 0x03 4 - Trap() - 0x05 5 - Report() - 0x06 6 - ReportResp() - 0x86 7 - TrapRepress() - 0x07 3. When method_select = 3 - All methods, the filter will apply for all the method values.	
04h + (8N)h	31:24	class	Class value to filter class = 0x0 indicates that the filter apply to all the classes.	
	15:0	attribute	Attribute value to filter attribute = 0x0 indicates that the filter apply to all the attributes.	

Table 2343 - SET_MAD_DEMUX Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.24.3MAD_IFC - Management Datagram Interface

Note: This command is supported only for physical function (PF) drivers and only when physical port is IB.

Note: Set operations supported only for vport0.

The MAD_IFC command takes a MAD packet from the Input Mailbox, processes it, and stores the response MAD in the Output Mailbox. Whether the command validates the MKey or BKey of the packet is controlled through the command Opcode Modifier. When the MAD interface is without MKey/Bkey validation, it is possible to locally query and control the HCA.

If the command fails to process the packet, it will return the BAD_PKT status while indicating that the Output Mailbox is empty, and that the packet is silently discarded by the local SMA/GSA.

For MADs with no response (for example, SubnTrapRepress) the return status will indicate OK (if packet processing passes), but the Output Mailbox will be invalid.

Table 2344 - MAD_IFC Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
remote_lid																								port								08h
																																0Ch
mad																																10h-10Ch

Table 2345 - MAD_IFC Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	opcode modifier: Bit 0 - ‘0’: Mad interface with MKey validation. ‘1’: Mad interface without MKey validation. Bit 1 - ‘0’: Mad interface with BKey validation. ‘1’: Mad interface without BKey validation.	
08h	31:16	remote_lid	Remote (source) LID of the MAD	
	7:0	port	Port of the MAD	
10h-10Ch	2048	mad	Request MAD Packet (256bytes)	

Table 2346 - MAD_IFC Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
response_mad_packet																																10h-10Ch

Table 2347 - MAD_IFC Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-10Ch	2048	response_mad_packet	Response MAD Packet (256bytes)	

29.25 SRQ Commands

Note: This section is deprecated from ISSI==1 and replaced with RMP and XRC_SRQ commands.

The following commands are used for controlling SRQ resources (SRQs are discussed in detail in “Shared Receive Queue (SRQ)” on page 365).

Table 2348 - SRQ Commands Overview

Mnemonic	Command	Description	Reference
CRE- ATE_SRQ	Allocate SRQ	Creates new SRQ	Section 29.25.1, on page 2103
DESTROY_S RQ	Destroy SRQ	Destroys an SRQ and enables the device to release the related resources	Section 29.25.2, on page 2105
QUERY_SRQ	Query SRQ	Retrieves a snapshot of the current SRQC entry	Section 29.25.3, on page 2106
RESIZE_SRQ	Resize SRQ	Resizes an existing SRQ	Section 22.19.4, on page 610
ARM_RQ	Arm SRQ/XRQ	Arms an SRQ/XRQ by setting its lwm field to a non-zero value	Section 29.25.4, on page 2108

29.25.1 CREATE_SRQ – Create SRQ

CREATE_SRQ command creates new SRQ. The command takes the SRQC entry from the input mailbox and uses it for new SRQ. CREATE_SRQ also transfers the physical pages of the SRQ buffer.

Table 2349 - CREATE_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																srqn																08h
																																0Ch
srq_context (See Table 112, “SRQ Context Layout,” on page 367)																																10h-4Ch
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
																																58h-10Ch

Table 2349 - CREATE_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pas[0]																															110h-114h	
pas[1]																															118h-11Ch	
...																															...	

Table 2350 - CREATE_SRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	srqn		
10h-4Ch	512	srq_context (See Table 112, “SRQ Context Layout,” on page 367)		
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803).	
58h-10Ch	1472	reserved		
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)		

Table 2351 - CREATE_SRQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								srqn																								08h

Table 2351 - CREATE_SRQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch

Table 2352 - CREATE_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	srqn	srq number	
0Ch	31:0	reserved		

29.25.2 DESTROY_SRQ – Destroy an SRQ

The DESTROY_SRQ command destroys an SRQ and enables the device to release the related resources.

Table 2353 - DESTROY_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									srqn																		08h					
																																0Ch

Table 2354 - DESTROY_SRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	[EMC] 0 - MTTs are destroyed automatically 1 - MTTs are destroyed using the DESTROY_MTT command	
08h	23:0	srqn	SRQ number	
0Ch	31:0	reserved		

Table 2355 - DESTROY_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																08h-0Ch

Table 2356 - DESTROY_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.25.3QUERY_SRQ – Query SRQ

The QUERY_SRQ command retrieves a snapshot of the current SRQC entry. The command stores the snapshot in the output mailbox in the software format. Note that the SRQC state and values are not affected by the QUERY_SRQ command.

The QUERY_SRQ command is for debug purposes only.

Table 2357 - QUERY_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									srqn																		08h					
																																0Ch

Table 2358 - QUERY_SRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2358 - QUERY_SRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	23:0	srqn	srq number	
0Ch	31:0	reserved		

Table 2359 - QUERY_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												04h				
																												08-0Ch				
srq_context (See Table 112, “SRQ Context Layout,” on page 367)																												10h-4Ch				
e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																												50h-54h				
reserved																												58h-10Ch				
pas[0]																												110h-114h				
pas[1]																												118h-11Ch				
...																												...				

Table 2360 - QUERY_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	srq_context (See Table 112, “SRQ Context Layout,” on page 367)		

Table 2360 - QUERY_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
50h-54h	64	e_mtt_pointer (See Table 1802, “e-mtt_pointer Layout,” on page 1864)	Pointer to MTTs in ICM. . Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803).	
58h-10Ch	1472	reserved		
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)		

29.25.4ARM_RQ

The ARM_RQ command is used to set the Limit Water Mark of the SRQ/XRQ.

Table 2361 - ARM_RQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																srqn ¹ _or_xrqn																08h
																lwm																0Ch

Table 2362 - ARM_RQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0x1: SRQ 0x2: XRQ	
08h	23:0	srqn ¹ _or_xrqn	SRQ/XRQ number	
0Ch	15:0	lwm	Limit water mark 0x0 will disarm the RQ,	

Table 2363 - ARM_RQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																															004h	
																															08h-0Ch	

Table 2364 - ARM_RQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.26 XRC_SRQ Commands

The following commands are used for controlling XRC_SRQ resources([TBD - fix section for XRC SRQs are discussed in detail in “Shared Receive Queue \(SRQ\)” on page 365](#)).

Table 2365 - XRC_SRQ Commands Overview

Mnemonic	Command	Description	Reference
CREATE_X-RC_SRQ	Allocate XRC_SRQ	Creates new XRC_SRQ	Section 29.26.1, on page 2110
DESTROY_X-RC_SRQ	Destroy XRC_SRQ	Destroys an XRC_SRQ and enables the device to release the related resources	Section 29.26.2, on page 2112
QUERY_X-RC_SRQ	Query XRC_SRQ	Retrieves a snapshot of the current XRC_SRQC entry	Section 29.26.3, on page 2113
ARM_X-RC_SRQ	Arm XRC_SRQ	Arms an XRC_SRQ by setting its lwm field to a non-zero value	Section 29.26.4, on page 2115

29.26.1 CREATE_XRC_SRQ – Create XRC_SRQ

CREATE_XRC_SRQ command creates a new XRC_SRQ. The command takes the XRC_SRQC entry from the input mailbox and uses it for the new XRC_SRQ. CREATE_XRC_SRQ also transfers the physical pages of the XRC_SRQ buffer.

Table 2366 - CREATE_XRC_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrc_srqn																								08h
																																0Ch
xrc_srq_context (See Table 114, “XRC_SRQ Context Layout,” on page 370)																																10h-4Ch
e_mtt_pointer_or_xrc_srq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h
xrc_srq_umem_id																																58h
xrc_srq_umem_valid																																5Ch
																																60h-10Ch
	pas[0]																															110h-114h
	pas[1]																															118h-11Ch
	...																															...

Table 2367 - CREATE_XRC_SRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrc_srqn		
10h-4Ch	512	xrc_srq_context (See Table 114, “XRC_SRQ Context Layout,” on page 370)		
50h-54h	64	e_mtt_pointer_or_eqxrc_srq_umem_offset (See Table 1802, “e_mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803). If xrc_srq_umem_valid==1, then indicates umem offset of the CQ	
58h	31:0	xrc_srq_umem_id	UMEM id of the CQ. Valid only xrc_srq_umem_valid==1.	
5Ch	31	xrc_srq_umem_valid	Indicates if cq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)	Valid only when cq_umem_valid==0	

Table 2368 - CREATE_XRC_SRQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								xrc_srqn																								08h
																																0Ch

Table 2369 - CREATE_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	xrc_srqn	xrc_srqn number	
0Ch	31:0	reserved		

29.26.2 DESTROY_XRC_SRQ – Destroy an XRC_SRQ

The DESTROY_XRC_SRQ command destroys an XRC_SRQ and enables the device to release the related resources.

Table 2370 - DESTROY_XRC_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									xrc_srqn																		08h					
																																0Ch

Table 2371 - DESTROY_XRC_SRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	[EMC] 0: Automatically - MTTs are destroyed automatically 1: DESTROY_MTT - MTTs are destroyed using the DESTROY_MTT command	
08h	23:0	xrc_srqn	XRC_SRQ number	
0Ch	31:0	reserved		

Table 2372 - DESTROY_XRC_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h

Table 2372 - DESTROY_XRC_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																004h
																																08h-0Ch

Table 2373 - DESTROY_XRC_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.26.3 QUERY_XRC_SRQ – Query XRC_SRQ

The QUERY_XRC_SRQ command retrieves a snapshot of the current XRC_SRQC entry. The command stores the snapshot in the output mailbox in the software format. Note that the XRC_SRQC state and values are not affected by the QUERY_XRC_SRQ command.

The QUERY_XRC_SRQ command is for debug purposes only.

Table 2374 - QUERY_XRC_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrc_srqn																								08h
																																0Ch

Table 2375 - QUERY_XRC_SRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrc_srqn	xrc_srq number	

Table 2375 - QUERY_XRC_SRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	reserved		

Table 2376 - QUERY_XRC_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
status																																					00h
syndrome																																04h					
																																08-0Ch					
xrc_srq_context (See Table 114, “XRC_SRQ Context Layout,” on page 370)																																10h-4Ch					
e_mtt_pointer_or_xrc_srq_umem_offset (See Table 1802, “e-mtt_pointer Layout,” on page 1864)																																50h-54h					
xrc_srq_umem_id																																58h					
xrc_srq_umem_valid																																5Ch					
																																60h-10Ch					
	pas[0]																															110h-114h					
	pas[1]																															118h-11Ch					
	...																															...					

Table 2377 - QUERY_XRC_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 2377 - QUERY_XRC_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
10h-4Ch	512	xrc_srq_context (See Table 114, “XRC_SRQ Context Layout,” on page 370)		
50h-54h	64	e_mtt_pointer_or_xrc_srq_umem_offset (See Table 1802, “e_mtt_pointer Layout,” on page 1864)	If HCA_CAP.e bit is set: Pointer to MTTs in ICM. Used instead of pas when global e bit is set (See Table 1627, “SET_HCA_CAP Input Structure Layout,” on page 1803). If xrc_srq_umem_valid==1, then indicates umem offset of the CQ	
58h	31:0	xrc_srq_umem_id	UMEM id of the CQ. Valid only xrc_srq_umem_valid==1.	
5Ch	31	xrc_srq_umem_valid	Indicates if cq buffer is given directly by PAS list or by umem_id and umem_offset. When 0, PAS list. When 1, umem_offset and umem_offset ars valid	
110h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)		

29.26.4 ARM_XRC_SRQ – Arm XRC SRQ

The ARM_XRC_SRQ command is used to set the Limit Water Mark (lwm) of the XRC_SRQ.

Table 2378 - ARM_XRC_SRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrc_srqn																								08h
																lwm																0Ch

Table 2379 - ARM_XRC_SRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0x1: XRC_SRQ	
08h	23:0	xrc_srqn	XRC_SRQ number	
0Ch	15:0	lwm	Limit water mark 0x0 will disarm the XRC_SRQ,	

Table 2380 - ARM_XRC_SRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																08h-0Ch

Table 2381 - ARM_XRC_SRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.27 XRQ Commands

The following commands are used for controlling XRQ resources as described in Section 8.21, “XRQ Context,” on page 509

Table 2382 - XRQ Commands Overview

Mnemonic	Command	Description	Reference
CRE-ATE_XRQ	Allocate XRQ	Creates new XRQ	Section 29.27.1, on page 2117
DESTROY_XRQ	Destroy XRQ	Destroys an XRQ and enables the device to release the related resources	Section 29.27.2, on page 2118

Table 2382 - XRQ Commands Overview (Continued)

Mnemonic	Command	Description	Reference
QUE- RY_XRQ	Query XRQ	Retrieves a snapshot of the current XRQ entry	Section 29.27.3, on page 2119

29.27.1 CREATE_XRQ – Create XRQ

CREATE_XRQ command creates a new XRQ.

Table 2383 - CREATE_XRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch
xrq_context (See Table 246, “XRQ Context Layout,” on page 510)																																10h-...

Table 2384 - CREATE_XRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-...	2624	xrq_context (See Table 246, “XRQ Context Layout,” on page 510)		

Table 2385 - CREATE_XRQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								xrqn																								08h

Table 2385 - CREATE_XRQ Output Structure Field Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch

Table 2386 - CREATE_XRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	23:0	xrqn	xrc_srq number	
0Ch	31:0	reserved		

29.27.2 DESTROY_XRQ – Destroy an XRQ

The DESTROY_XRQ command destroys an XRQ and enables the device to release the related resources.

Table 2387 - DESTROY_XRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
																																0Ch

Table 2388 - DESTROY_XRQ Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number	
0Ch	31:0	reserved		

Table 2389 - DESTROY_XRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2390 - DESTROY_XRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.27.3QUERY_XRQ – Query XRQ

The QUERY_XRQ command retrieves a snapshot of the current XRQ context.

Table 2391 - QUERY_XRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
																																0Ch

Table 2392 - QUERY_XRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	xrq number	

Table 2392 - QUERY_XRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	reserved		

Table 2393 - QUERY_XRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																04h				
																																08-0Ch				
xrq_context (See Table 246, “XRQ Context Layout,” on page 510)																																10h-...				

Table 2394 - QUERY_XRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	2624	xrq_context (See Table 246, “XRQ Context Layout,” on page 510)		

29.27.4MODIFY_XRQ – Modify XRQ

The MODIFY_XRQ command is used to modify XRQ context. Supported only when HCA_CAP.modify_xrq==1.

Table 2395 - MODIFY_XRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h

Table 2395 - MODIFY_XRQ Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								xrqn																							08h	
field_select																															0Ch	
xrq_context (See Table 246, “XRQ Context Layout,” on page 510)																															10h-...	

Table 2396 - MODIFY_XRQ input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	xrqn number	
0Ch	31:0	field_select	Field select indicates in QUERY which fields are modifiable (1 means modifiable), and in MODIFY which fields to modify.	
10h-...	2624	xrq_context (See Table 246, “XRQ Context Layout,” on page 510)		

Table 2397 - MODIFY_XRQ Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																															04h	
																															08-0Ch	

Table 2398 - MODIFY_XRQ Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.27.5 QUERY_XRQ_DC_PARAMS_ENTRY – Query XRQ DC PARAMS Entry

For supporting application level offload in XRQ on top of unconnected transport services, for example Dynamically Connected Transport (DC) the XRQ need to get the network parameters for creating communication with the source application requesting the XRQ services.

SW needs to provide a manner for building the send queue to be used for responding to the remote node in the transport service above therefore will require to have a allocate these parameters in the XRQ DC PARAMS TABLE by SET_XRQ_DC_PARAMS_ENTRY.

Table 2399 - QUERY_XRQ_DC_PARAMS_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
																																0Ch

Table 2400 - QUERY_XRQ_DC_PARAMS_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number that allocates the table.	

Table 2401 - QUERY_XRQ_DC_PARAMS_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																00h																								
syndrome																																04h																								
																																08h-0Ch																								

Table 2401 - QUERY_XRQ_DC_PARAMS_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
xrq_dc_params_entry (See Table 2403, “XRQ DC PARAMS ENTRY Layout,” on page 2123)																																10h-2C

Table 2402 - QUERY_XRQ_DC_PARAMS_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-2C	256	xr_q_dc_params_entry (See Table 2403, “XRQ DC PARAMS ENTRY Layout,” on page 2123)	xr_q_dc parameters entry.	

Table 2403 - XRQ DC PARAMS ENTRY Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset					
			ack_timeout									multi_path	mtu		pkey_table_index																		00h				
													cnak_reverse_sl					reverse_sl							sl			04h									
dc_access_key																																					08h-0Ch
																																					10h-1Ch

Note that the description of fields below similar to the description of same fields in QP context.

Table 2404 - XRQ DC PARAMS ENTRY Field Descriptions

Offset	Bits	Name	Description	Access
00h	28:24	ack_timeout		
	19	multi_path		
	18:16	mtu		
	15:0	pkey_table_index		
04h	19:16	cnak_reverse_sl		
	11:8	reverse_sl		
	3:0	sl		
08h-0Ch	64	dc_access_key		

29.27.6SET_XRQ_DC_PARAMS_ENTRY – SET XRQ DC PARAMS Entry

Table 2405 - SET_XRQ_DC_PARAMS_ENTRY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																xrqn																08h
																																0Ch
																xrq_dc_params_entry (See Table 2403, “XRQ DC PARAMS ENTRY Layout,” on page 2123)																10h-2C

Table 2406 - SET_XRQ_DC_PARAMS_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2406 - SET_XRQ_DC_PARAMS_ENTRY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	23:0	xrqn	XRQ number that allocates the table.	
10h-2C	256	xrq_dc_params_entry (See Table 2403, “XRQ DC PARAMS ENTRY Layout,” on page 2123)	xrq dc parameters entry.	

Table 2407 - SET_XRQ_DC_PARAMS_ENTRY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2408 - SET_XRQ_DC_PARAMS_ENTRY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.27.7 QUERY_XRQ_ERROR_PARAMS – Query XRQ Error Parameters

XRQ may induce an event upon error (XRQ ERROR EVENT, See Section 8.20.10, “XRQ Error Event,” on page 482), some of the event may carry the data associated within the events while other may require the software to query the parameters using the `xrq_error_handle` communicated in the event. After querying the error parameters, SW should call `RELEASE_XRQ_ERROR` as soon as possible to release the XRQ resources .

Table 2409 - QUERY_XRQ_ERROR_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				

Table 2409 - QUERY_XRQ_ERROR_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								xrqn																							08h	
								xrq_error_handle																							0Ch	

Table 2410 - QUERY_XRQ_ERROR_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number.	
0Ch	23:0	xrq_error_handle	XRQ error handle that reported in the XRQ_ERROR_EVENT	

Table 2411 - QUERY_XRQ_ERROR_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
xrq_error_params (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)																																10h-4C

Table 2412 - QUERY_XRQ_ERROR_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4C	512	xrq_error_params (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)	xrq error parameters	

Table 2413 - XRQ ERROR PARAMS Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								local_dct																								00h
								remote_dct																								04h
																remote_lid																08h
																																0Ch
remote_mac (See Table 33, “MAC Address Layout,” on page 293)																																10h-14h
remote_l3_addr																																18h-1Ch
																																20h-3Ch

Table 2414 - XRQ ERROR PARAMS Field Descriptions

Offset	Bits	Name	Description	Access
00h	23:0	local_dct	local DCT	
04h	23:0	remote_dct	Remote DCT	
08h	15:0	remote_lid	Remote LID (IB)	
10h-14h	64	remote_mac (See Table 33, “MAC Address Layout,” on page 293)	REMOTE MAC address (ROCE)	
18h-1Ch	64	remote_l3_addr	Remote L3 address. Can be 128 bits gid, 128 bits of IPV6 or 32 bits ip address. Note that for IPv4, bits[127:32] are reserved.	

29.27.8 RELEASE_XRQ_ERROR – Release XRQ Error

XRQ may induce an event upon error (XRQ ERROR EVENT, See Section 8.20.10, “XRQ Error Event,” on page 482), some of the event may carry the data associated within the events while other may require the software to query the parameters using the `xrq_error_handle` communicated in the event. After querying the error parameters, SW should call `RELEASE_XRQ_ERROR` as soon as possible to release the XRQ resources. In this command SW should specify the parameters of the jobs that should be flushed before releasing the XRQ and returning it back to work.

Table 2415 - RELEASE_XRQ_ERROR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																08h								
								xrq_error_handle																0Ch								
xrq_error_params (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)																																10h-4C
xrq_error_params_mask (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)																																50h-8C

Table 2416 - RELEASE_XRQ_ERROR Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number.	
0Ch	23:0	xrq_error_handle	XRQ error handle that reported in the XRQ_ERROR_EVENT	
10h-4C	512	xrq_error_params (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)	XRQ error parameters of the jobs that should be flushed. This valid parameters is according to <code>xrq_error_params_mask</code>	
50h-8C	512	xrq_error_params_mask (See Table 2413, “XRQ ERROR PARAMS Layout,” on page 2127)	xrq error parameters Mask.	

Table 2417 - RELEASE_XRQ_ERROR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																04h
																																08h-0Ch

Table 2418 - RELEASE_XRQ_ERROR Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.28 NVMe-oF Commands

29.28.1 CREATE_NVMF_BACKEND_CONTROLLER – Create NVMe-oF Backend Controller

The CREATE_NVMF_BACKEND_CONTROLLER command is used to create a new backend controller in a specific XRQ.

Table 2419 - CREATE_NVMF_BACKEND_CONTROLLER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
																																0Ch
nvmf_backend_controller (See Table 2423, “NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130)																																10h-...

Table 2420 - CREATE_NVMF_BACKEND_CONTROLLER input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number to attach the namespace to it	
10h-...	960	nvmf_backend_controller (See Table 2423, “NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130)	NVMe-oF controller context	

Table 2421 - CREATE_NVMF_BACKEND_CONTROLLER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																backend_controller_id																08h
																																0Ch

Table 2422 - CREATE_NVMF_BACKEND_CONTROLLER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	15:0	backend_controller_id	NVMe backend controller identifier	

Table 2423 - NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
						cq_page_offset															sq_page_offset										00h	
											cq_log_page_size															sq_log_page_size					04h	
																																08h

Table 2423 - NVMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
initial_cqh_db_value																initial_sqt_db_value																0Ch
																												log_cmd_time-out_us	10h			
																													14-2Ch			
cq_h_dbr_addr																													30h-34h			
sqt_dbr_addr																													38h-3Ch			
																													40h-6Ch			
pas[0] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)																													70h-74h			
pas[1] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)																													78h-7Ch			
...																													80h-..h			

Table 2424 - NVMF_BACKEND_CONTROLLER_CONTEXT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	25:16	cq_page_offset	Page offset in granularity of $2^{\log\_memory\_page\_size}$	
	9:0	sq_page_offset	Page offset in granularity of $2^{\log\_memory\_page\_size}$	
04h	20:16	cq_log_page_size	Log (base 2) of page size of NVMe cq in units of 4Kbyte.	
	4:0	sq_log_page_size	Log (base 2) of page size of NVMe sq in units of 4Kbyte.	
0Ch	31:16	initial_cqh_db_value	The initial completion queue head doorbell value	
	15:0	initial_sqt_db_value	The initial send queue tail doorbell value	

Table 2424 - NVMF_BACKEND_CONTROLLER_CONTEXT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	4:0	log_cmd_timeout_us	Log(base2) of timeout (in useconds) for a command that submitted to the NVMe backend device until command completion was seen. Device sends an error completion back to the initiator upon this timeout. An NVMe completion that will arrive after the timeout will be ignored (and counted in comp_after_timeout counter). Must be between HCA_CAP.log_min_cmd_timeout and HCA_CAP.log_max_cmd_timeout 0 means that the device uses its default time out value.	
30h-34h	64	cqh_dbr_addr	Physical address bits of NVMe completion queue head doorbell record. Must be aligned to 4 bytes.	
38h-3Ch	64	sqt_dbr_addr	Physical address bits of NVMe submission queue tail doorbell record. Must be aligned to 4 bytes.	
70h-..	64	pas[...] (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)	Array of physical address structure (PAS) of the CQ and SQ. First the PAS of CQ, followed by PAS of SQ. Note that each PAS list must be aligned to 64 Bytes.	

29.28.2 DESTROY_NVMF_BACKEND_CONTROLLER – Destroy NVMe-oF Backend Controller

The DESTROY_NVMF_BACKEND_CONTROLLER command is used to destroy backend controller in a specific XRQ.

Table 2425 - DESTROY_NVMF_BACKEND_CONTROLLER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
								xrqn																				08h				
																backend_controller_id												0Ch				

Table 2426 - DESTROY_NVMF_BACKEND_CONTROLLER input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2426 - DESTROY_NVMMF_BACKEND_CONTROLLER input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number	
0Ch	15:0	backend_controller_id	NVMe backend controller identifier	

Table 2427 - DESTROY_NVMMF_BACKEND_CONTROLLER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																00h																								
syndrome																																04h																								
																																08-0Ch																								

Table 2428 - DESTROY_NVMMF_BACKEND_CONTROLLER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.28.3QUERY_NVMMF_BACKEND_CONTROLLER – Query NVMe-oF Backend Controller

The QUERY_NVMMF_BACKEND_CONTROLLER command is used to query a context of backend controller in a specific XRQ

Table 2429 - QUERY_NVMMF_BACKEND_CONTROLLER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
																backend_controller_id																0Ch

Table 2430 - QUERY_NVMMF_BACKEND_CONTROLLER input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number	
0Ch	15:0	backend_controller_id	NVMe backend controller identifier	

Table 2431 - QUERY_NVMMF_BACKEND_CONTROLLER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
nvmmf_backend_controller (See Table 2423, “NVMMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130)																																10h-...

Table 2432 - QUERY_NVMMF_BACKEND_CONTROLLER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	960	nvmmf_backend_controller (See Table 2423, “NVMMF_BACKEND_CONTROLLER_CONTEXT Input Structure Layout,” on page 2130)	NVMe-oF controller context	

29.28.4 ATTACH_NVMMF_NAMESPACE – Attach NVMe-oF Namespace

The ATTACH_NVMMF_NAMESPACE command is used to attach a new NVMe namespace to a specific XRQ.

Table 2433 - ATTACH_NVMF_NAMESPACE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
frontend_namespace																																0Ch
backend_namespace																																10h
lba_data_size																backend_controller_id																14h
																metadata_size																18h
																																1Ch-3Ch

Table 2434 - ATTACH_NVMF_NAMESPACE input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number to attach the namespace to.	
0Ch	31:0	frontend_namespace	Front-end namespace to be attached to XRQ.	
10h	31:0	backend_namespace	NVMe backend namespace to be used to handle the attached front-end namespace	
14hh	31:16	lba_data_size	LBA data size in granularity of Bytes. Minimal allowed 512B. Must be power of 2.	
	15:0	backend_controller_id	NVMe backend controller identifier to be used to handle the attached front-end namespace.	
18h	15:0	metadata_size	The number of metadata bytes per LBA. 0 means no metadata. The maximum support size is HCA_CAP.max_lba_metadata_size.	

Table 2435 - ATTACH_NVMF_NAMESPACE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																04h
																																08-0Ch

Table 2436 - ATTACH_NVMF_NAMESPACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.28.5 DETACH_NVMF_NAMESPACE – Detach NVMe-oF Namespace

The DETACH_NVMF_NAMESPACE command is used to detach an NVMe namespace from a specific XRQ.

Table 2437 - DETACH_NVMF_NAMESPACE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
frontend_namespace																																0Ch

Table 2438 - DETACH_NVMF_NAMESPACE input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number	
0Ch	31:0	frontend_namespace	Front-end namespace to be detached from the XRQ.	

Table 2439 - DETACH_NVMF_NAMESPACE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch

Table 2440 - DETACH_NVMF_NAMESPACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.28.6 QUERY_NVMF_NAMESPACE_CONTEXT – Query NVMF NameSpace Context

This command is supported only when HCA_CAP.frontend_namespace_context == 1.

Table 2441 - QUERY_NVMF_NAMESPACE_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								xrqn																								08h
frontend_namespace																																0Ch

Table 2442 - QUERY_NVMF_NAMESPACE_CONTEXT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	xrqn	XRQ number	

Table 2442 - QUERY_NVMF_NAMESPACE_CONTEXT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	frontend_namespace	Frontend namespace	

Table 2443 - QUERY_NVMF_NAMESPACE_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												04h				
																												08-0Ch				
nvmf_frontend_namespace_context (See Table 2445, “NVMF_NAMESPACE_CONTEXT Input Structure Layout,” on page 2138)																												10h-...				

Table 2444 - QUERY_NVMF_NAMESPACE_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	896	nvmf_frontend_namespace_context (See Table 2445, “NVMF_NAMESPACE_CONTEXT Input Structure Layout,” on page 2138)	Nvmf frontend namespace context	

Table 2445 - NVMF_NAMESPACE_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
num_read_cmd																																00h-04h
num_read_blocks																																08h-0Ch
num_write_cmd																																10h-14h

Table 2445 - NVMF_NAMESPACE_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
num_write_blocks																																18h-1Ch
num_write_inline_cmd																																20h-24h
num_flush_cmd																																28h-2Ch
num_error_cmd																																30h-34h
num_backend_error_cmd																																38h-3Ch
																																40h-6Ch

Table 2446 - NVMF_NAMESPACE_CONTEXT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h-04h	64	num_read_cmd	Number of Read Commands. Note that only the 32 LSB bits are valid.	
08h-0Ch	64	num_read_blocks	Number of Read Blocks. Note that only the 32 LSB bits are valid.	
10h-14h	64	num_write_cmd	Number of Write Commands. Note that only the 32 LSB bits are valid.	
18h-1Ch	64	num_write_blocks	Number of written Blocks. Note that only the 32 LSB bits are valid.	
20h-24h	64	num_write_inline_cmd	Number of Write Commands with inline. Note that only the 32 LSB bits are valid.	
28h-2Ch	64	num_flush_cmd	Number of Flush Commands. Note that only the 32 LSB bits are valid.	
30h-34h	64	num_error_cmd	Number of Error Commands.(Bad or unsupported) Note that only the 32 LSB bits are valid.	
38h-3Ch	64	num_backend_error_cmd	Number of commands that got bad status completion from the backend NVMe Note that only the 32 LSB bits are valid.	

29.29 QUERY_ESW_FUNCTIONS - Query e-Switch Functions

This command can be used by eswitch manager to identify and retrieve information about the functions (Host PF, VFs and SFs) that are connected to the eswitch.

As this information can be changed on the fly, SW of eswitch manager needs to register to ESW_FUNCTIONS_CHANGED event (See Section 8.20.14.4, “ESW_FUNCTIONS_CHANGED Event,” on page 503).

The event is generated if any of the following occurs:

A vport is enabled/disabled in any of the following events:

- The PF is enabled/disabled
 - In this case the PF vport and all its VF/SF vports are enabled/disabled at once
- SR-IOV num_vfs change
 - In this case all VF vports are enabled/disabled at once
- ALLOC/DEALLOC_SF command

In response, the eswitch manager driver should issue a QUERY_ESW_FUNCTIONS command to retrieve the enabled eswitch vports. If host_pf_enabled=0 then all external host vports are disabled. Otherwise, the enabled VF vports are specified by the host_num_of_vfs field and the enabled SF vports are specified by the sf_enable bitmap.”

The command and the ESW_FUNCTIONS_CHANGED event are supported when HCA_CAP.esw_functions_changed==1, or when INIT_SEGMENT.embedded_cpu==1.

Table 2447 - QUERY_ESW_FUNCTIONS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2448 - QUERY_ESW_FUNCTIONS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2449 - QUERY_ESW_FUNCTIONS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
host_params_context (See Table 2451, “HOST_PARAMS Context Layout,” on page 2142)																																10h-4Ch
																																50h-7Ch
sf_enable[0]																																80h-84h
sf_enable[1]																																88h-8Ch
...																																

Table 2450 - QUERY_ESW_FUNCTIONS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-4Ch	512	host_params_context (See Table 2451, “HOST_PARAMS Context Layout,” on page 2142)	External host parameters.	
80h-...	64	sf_enable[...]	Bit per Sub-function. indicates if SF is enabled.	

Table 2451 - HOST_PARAMS Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
host_number														host_pf_vhca_id_valid	host pf disabled	host_num_vfs																00h
host_total_vfs																host_pci_bus																04h
host_pf_vhca_id																host_pci_device																08h
																host_pci_function																0Ch
																																10h-3Ch

Table 2452 - HOST_PARAMS Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	host_number	Host number (identifier)	
	17	host_pf_vhca_id_valid	If set, indicates that host_pf_vhca_id is valid.	
00h	16	host_pf_disabled	If set, host pf is disabled. The host PF is assumed to be enabled following its ENABLE_HCA command until it is disabled by the DISABLED_HCA command. Otherwise, the PF is assumed to be disabled. A PF reset event also moves it to a disabled state. A reset event in this context may be a PCI configuration cycle setting the FLR bit, a PCI memory write transaction to the <i>nic_interface</i> field of the initialization segment or a PCI bus reset. While a PF is disabled, all e-Switch vports in its domain (its vport and its VF/SF vports) are assumed to be disabled. While a PF is enabled, a VF e-Switch vport is assumed to be enabled for all VFs in the range of SR-IOV num_vfs and a SF e-Switch vport is assumed to be enabled upon ALLOC_SF command and disabled upon DEALLOC_SF command. Whenever one or more e-Switch vports are enabled/disabled, the device issues an ESW_FUNCTIONS_CHANGED event.	
	15:0	host_num_vfs	Number of virtual functions on the external host side.	

Table 2452 - HOST_PARAMS Context Field Descriptions (Continued)

Offset	Bits	Name	Description	Access
04h	31:16	host_total_vfs	Total number of virtual functions supported by the device on the external host side. 0 means the field is invalid.	
	15:0	host_pci_bus	The PCI bus (B part of the BDF) of the host function that controlled by the EC.	
08h	31:16	host_pf_vhca_id	VHCA ID of external host PF. Valid only when host_pf_vhca_id_valid is set.	
	15:0	host_pci_device	The PCI device (D part of the BDF) of the host function that controlled by the EC.	
0Ch	15:0	host_pci_function	The PCI physical function (F part of the BDF) of the host function that controlled by the EC.	

29.30 DCT Commands

The following commands are used for controlling DCT resources (DCTs are discussed in detail in “DCT - Dynamically Connected Target” on page 279).

Table 2453 - DCT Commands Overview

Command	Description	Reference
CREATE_DCT	Creates new DCT	Section 29.30.1, “CREATE_DCT – Create DCT,” on page 2144
DESTROY_DCT	Destroys an DCT and enables the device to release the related resources.	Section 29.30.2, “DESTROY_DCT – Destroying a DCT,” on page 2145
QUERY_DCT	Retrieves a snapshot of the current DCT Context	Section 29.30.3, “QUERY_DCT – Query DCT,” on page 2146
DRAIN_DCT	Moves DCT to Drain state, which will close all current connection and prevent opening new connections	Section 29.30.4, “DRAIN_DCT – Draining a DCT,” on page 2147
ARM_DCT_FOR_KEY_VIOLATION	Arms DCT to create event in case of access key violation	Section 29.30.5, “ARM_DCT_FOR_KEY_VIOLATION – Arms DCT for Key Violation,” on page 2148

29.30.1 CREATE_DCT – Create DCT

CREATE_DCT command creates new DCT. The command takes the DCT entry from the input mailbox and uses it for new DCT.

Table 2454 - CREATE_DCT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																																0Ch
dct_context (See Table 25, “DCT Context Layout,” on page 279)																																10h-7Ch

Table 2455 - CREATE_DCT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h-7Ch	896	dct_context (See Table 25, “DCT Context Layout,” on page 279)		

Table 2456 - CREATE_DCT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
								dctn																								08h
ece																																0Ch

Table 2457 - CREATE_DCT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 2457 - CREATE_DCT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
08h	23:0	dctn	DCT number	
0Ch	31:0	ece	Enhanced Connection Establishment provided by device. 0 means invalid. Supported when HCA_CAP.ece = 1	

29.30.2 DESTROY_DCT – Destroying a DCT

The DESTROY_DCT command destroys an DCT and enables the device to release the related resources. It is recommended to use this command as explain in Section 16.2, “DC Control Objects,” on page 872.

Table 2458 - DESTROY_DCT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									dctn																		08h					
																																0Ch

Table 2459 - DESTROY_DCT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	dctn	DCT number	
0Ch	31:0	reserved		

Table 2460 - DESTROY_DCT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 2460 - DESTROY_DCT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 2461 - DESTROY_DCT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.30.3 QUERY_DCT – Query DCT

The QUERY_DCT command retrieves a snapshot of the current DCT entry. The command stores the snapshot in the output mailbox in the software format. Note that the DCT state and values are not affected by the QUERY_DCT command.

The QUERY_DCT command is for debug purposes only.

Table 2462 - QUERY_DCT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																dctn																08h
																																0Ch

Table 2463 - QUERY_DCT input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	dctn	DCT number	
0Ch	31:0	reserved		

Table 2464 - QUERY_DCT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08-0Ch
dct_context (See Table 25, “DCT Context Layout,” on page 279)																																10h-7Ch

Table 2465 - QUERY_DCT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-7Ch	896	dct_context (See Table 25, “DCT Context Layout,” on page 279)		

29.30.4DRAIN_DCT – Draining a DCT

The DRAIN_DCT command moves DCT to the Draining state, its requested to call this command before destroying DCT. This command will disconnect all connections related the drained DCT and prevent establishing new connections.

After disconnecting all connections, the device will generate an event and move DCT to a Drained state. See Section 16.2, “DC Control Objects,” on page 872.

Table 2466 - DRAIN_DCT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
									dctn																08h							
																																0Ch

Table 2467 - DRAIN_DCT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	dctn	DCT number	
0Ch	31:0	reserved		

Table 2468 - DRAIN_DCT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																08h-0Ch

Table 2469 - DRAIN_DCT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.30.5 ARM_DCT_FOR_KEY_VIOLATION – Arms DCT for Key Violation

HW can generate an async event (See Section 8.20.8.1, “DCT Events,” on page 480) in case of dc access key violation, i.e when a connect arrive with dc_access_key different form the one in the DCT.

To enable generating this event, driver should arm the DCT using ARM_DCT_FOR_KEY_VIOLATION command.

Table 2470 - ARM_DCT_FOR_KEY_VIOLATION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h

Table 2470 - ARM_DCT_FOR_KEY_VIOLATION Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
								dctn																							08h	
																															0Ch	

Table 2471 - ARM_DCT_FOR_KEY_VIOLATION Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	dctn	DCT number	
0Ch	31:0	reserved		

Table 2472 - ARM_DCT_FOR_KEY_VIOLATION Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2473 - ARM_DCT_FOR_KEY_VIOLATION Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.30.6 CREATE_DCT_LAG – Create DCT LAG

When HCA_CAP.lag_dct=0x1(SW_ASSISTED), SW needs to manage two device sub DCTs per logical DCT (that is visible to the verbs consumer application and found on the network wire).

Upon logical DCT creation, SW should create two sub DCTs; one DCT per port.

The content of dct_context struct of the CREATE_DCT input mailbox of both sub DCTs must be the same.

A DCT_Access_violation asynchronous event reported on any of the device sub DCTs should be reported by SW to the verbs consumer application as affiliated with the single logical DCT.

Upon logical DCT destruction, SW must destroy the two device sub DCTs in reverse order of their creation.

Table 2474 - CREATE_DCT_LAG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																																0Ch
								dctn0																								10h
								dctn1																								14h
																																18h-3Ch

Table 2475 - CREATE_DCT_LAG Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
10h	23:0	dctn0	First DCT Number.	
14h	23:0	dctn1	Second DCT Number.	

Table 2476 - CREATE_DCT_LAG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2477 - CREATE_DCT_LAG Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.30.7SET_DC_CNAK_TRACE – Set DC CNAK Trace

SET_DC_CNAK_TRACE used to enable/disable DC cnak trace and set the physical address where device dump the DC CNAK TRACE TABLE.

This feature supported only when HCA_CAP.dc_cn timer==1 (See Table 1571, “HCA Capabilities Layout,” on page 1671).

Table 2478 - SET_DC_CNAK_TRACE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
enable																																08h
																																0Ch-34h
pas (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)																																38h-3Ch

Table 2479 - SET_DC_CNAK_TRACE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	enable	Enable/Disable DC CNAK TRACE 0: disable 1: enable	
38h-3Ch	64	pas (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)	Physical address where device dumps the DC CNAK TRACE TABLE.	

Table 2480 - SET_DC_CNAK_TRACE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2481 - SET_DC_CNAK_TRACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.30.8 QUERY_DC_CNAK_TRACE – Query DC CNAK Trace

Table 2482 - QUERY_DC_CNAK_TRACE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2483 - QUERY_DC_CNAK_TRACE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2484 - QUERY_DC_CNAK_TRACE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
enable																																08h
																																0Ch-34h
pas (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)																																38h-3Ch

Table 2485 - QUERY_DC_CNAK_TRACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2485 - QUERY_DC_CNAK_TRACE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31	enable	DC CNAK TRACE is Enabled/Disabled. 0: disable 1: enable	
38h-3Ch	64	pas (See Table 1561, “Physical Address Structure (PAS) Layout,” on page 1665)	Physical address where device dump the CNAK TRACE TABLE.	

29.31 Vport Commands

The following commands are used to control and manage virtual ports state and context. Vports and more information about using these commands are discussed in Section 3.1, “Port Model Overview,” on page 180.

Table 2486 - Vport Commands Overview

Command	Description	Reference
QUERY_VPORT_STATE	Query Vport and Uplink state.	Section 29.31.1, “QUERY_VPORT_STATE – Query Vport State,” on page 2155
MODIFY_VPORT_STATE	Modify Vport and Uplink state	Section 29.31.2, “MODIFY_VPORT_STATE – Modify Vport State,” on page 2157
QUERY_ESW_VPORT_CONTEXT	Query E-switch Vport Context	Section 29.31.3, “QUERY_ESW_VPORT_CONTEXT – Query E-switch Vport Context,” on page 2158
MODIFY_ESW_VPORT_CONTEXT	Modify E-switch Vport Context	Section 29.31.4, “MODIFY_ESW_VPORT_CONTEXT – Modify E-switch Vport Context,” on page 2160
QUERY_NIC_VPORT_CONTEXT	Query NIC Vport Context	Section 29.31.5, “QUERY_NIC_VPORT_CONTEXT – Query NIC Vport Context,” on page 2162
MODIFY_NIC_VPORT_CONTEXT	Modify NIC Vport Context	Section 29.31.6, “MODIFY_NIC_VPORT_CONTEXT – Modify NIC Vport Context,” on page 2164
QUERY_VNIC_ENV	Query NIC/HCA Vport statistics, status and monitors	Section 29.34.3, “QUERY_VNIC_ENV – Query vNIC Environment,” on page 2181

29.31.1 QUERY_VPORT_STATE – Query Vport State

This command is used to get information about the nic_vports, esw_vports and Uplinks.

Table 2487 - QUERY_VPORT_STATE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport																vport_number																08h
																																0Ch

Table 2488 - QUERY_VPORT_STATE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: vnic_vport 1: esw_vport 2: uplink	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. allowed only if HCA - CAP.vport_group_manager. For nic_vport and uplink: must be 0.	
	15:0	vport_number	~~For nic_vport and uplink: must be 0.~~ For esw_vport: number of other esw_vport on same e-switch - valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	

Table 2489 - QUERY_VPORT_STATE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 2489 - QUERY_VPORT_STATE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																008h
max_tx_speed																								admin_state				state				0Ch

Table 2490 - QUERY_VPORT_STATE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:16	max_tx_speed	The actual speed available for this function in granularity of 100 Mb/s. Value 0x0 indicates this field is not supported.	
	7:4	admin_state	The state required by SW 0x0: down - set port to down state 0x1: up - set port to up state 0x2: follow - set port to follow state (unavailable for nic_vport) others - reserved	
	3:0	state	port state 0x0: down - port is down 0x1: up - port is up others - reserved	

29.31.2 MODIFY_VPORT_STATE – Modify Vport State

Table 2491 - MODIFY_VPORT_STATE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport																vport_number																08h
																							admin_state							0Ch		

Table 2492 - MODIFY_VPORT_STATE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: nic_vport 1: esw_vport 2: Uplink	
08h	31	other_vport	0:myvport- access my vport. vport_number is reserved. 1:othervport- access another vport determined by vport_number on my e-switch. allowed only if HCA - CAP.vport_group_managerswitch_manager. For nic_vport and uplink: must be 0.	
	15:0	vport_number	~~For nic_vport and uplink: must be 0.~~ For esw_vport: number of other esw_vport on same e-switch - valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	7:4	admin_state	0x0: down - set port to down state 0x1: up - set port to up state 0x2: follow - set port to follow state (unavailable for nic_vport) others - reserved. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	

Table 2493 - MODIFY_VPORT_STATE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																008h-0ch

Table 2494 - MODIFY_VPORT_STATE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	31:0	reserved		

29.31.3 QUERY_ESW_VPORT_CONTEXT – Query E-switch Vport Context

This command is used to query esw vports context and can be performed only by vport_group_manager (HCA_CAP.vport_group_manager==1) that is also the eswitch manager (HCA_CAP.eswitch_manager==1).

Table 2495 - QUERY_ESW_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport																vport_number																08h
																															0Ch	

Table 2496 - QUERY_ESW_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch.	
	15:0	vport_number	Number of other esw_vport on same e-switch - can be used just by PF which access esw_vports of its VFs. Valid only if other_vport =1. For more details See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	31:0	reserved		

Table 2497 - QUERY_ESW_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
esw_vport_context (See Table 27, “Esw_vport Context Layout,” on page 284)																																10h-10C

Table 2498 - QUERY_ESW_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-10Ch	2048	esw_vport_context (See Table 27, “Esw_vport Context Layout,” on page 284)	esw_vport context	

29.31.4 MODIFY_ESW_VPORT_CONTEXT – Modify E-switch Vport Context

This command used to modify esw vports context and can be performed only by vport_group_manager (HCA_CAP.vport_group_manager==1) that is also the eswitch manager (HCA_CAP.eswitch_manager==1).

Table 2499 - MODIFY_ESW_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport																vport_number																08h
																															0Ch	
field_select																															10h-10Ch	
esw_vport_context (See Table 27, “Esw_vport Context Layout,” on page 284)																															10h-10Ch	

Table 2500 - MODIFY_ESW_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch.	
	15:0	vport_number	Number of other esw_vport on same e-switch - can be used just by PF which access esw_vports of its VFs.valid only if other_vport =1. For more details See Section 14.1.2, “Virtual Ports,” on page 839.	

Table 2500 - MODIFY_ESW_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	field_select	Selects fields out of esw_vport context to be modified. fields definition bit 0: svlan_strip - Valid field is vport_svlan_strip bit 1: cvlan_strip - Valid field is vport_cvlan_strip bit 2: svlan_insert - Valid fields are vport svlan_insert, svlan_cfi, svlan_pcp and svlan_id. bit 3: cvlan_insert - Valid fields are vport cvlan_insert, cvlan_cfi, cvlan_pcp and cvlan_id. bit 4: fdb_to_vport_reg_c_id	
10h-10Ch	2048	esw_vport_context (See Table 27, “Esw_vport Context Layout,” on page 284)	esw_vport context	

Table 2501 - MODIFY_ESW_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																008h-0ch

Table 2502 - MODIFY_ESW_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	31:0	reserved		

29.31.5 QUERY_NIC_VPORT_CONTEXT – Query NIC Vport Context

Table 2503 - QUERY_NIC_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport																vport_number																08h
						allowed_list_type																										0Ch

Table 2504 - QUERY_NIC_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager =1	
	15:0	vport_number	Number of other esw_vport on same e-switch - valid only if other_vport =1. For more details See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	26:24	allowed_list_type	Indicates which allowed list to query. Definition of this field is described in nic_vport_context. See Table 29, “NIC_Vport Context Layout,” on page 287.	

Table 2505 - QUERY_NIC_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
																syndrome																04h
																																08h-0Ch

Table 2505 - QUERY_NIC_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
nic_vport_context (See Table 29, “NIC_Vport Context Layout,” on page 287)																																10h-...

Table 2506 - QUERY_NIC_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	2112	nic_vport_context (See Table 29, “NIC_Vport Context Lay- out,” on page 287)	nic_vport context. See Table 29, “NIC_Vport Context Lay- out,” on page 287	

29.31.6 MODIFY_NIC_VPORT_CONTEXT – Modify NIC Vport Context

Table 2507 - MODIFY_NIC_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
other_vport																vport_number																08h	
field_select																																0Ch	
																																0Ch-0FCh	
nic_vport_context (See Table 29, “NIC_Vport Context Layout,” on page 287)																																100h-...	

Table 2508 - MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number on my e-switch. Allowed only if HCA_CAP.vport_group_manager =1	
	15:0	vport_number	Number of other esw_vport on same e-switch - valid only if other_vport =1. For more details See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	31:0	field_select	Selects fields out of nic_vport context to be modified. fields definition The bitmask defined in MODIFY_NIC_VPORT_CONTEXT Bitmask, See Table 35, “MODIFY_NIC_VPORT_CONTEXT Bitmask,” on page 293.	

Table 2508 - MODIFY_NIC_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
100h- ...	2112	nic_vport_context (See Table 29, “NIC_Vport Context Lay- out,” on page 287)	nic_vport context	

Table 2509 - MODIFY_NIC_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																				004h
																																				008h-0ch

Table 2510 - MODIFY_NIC_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h- 0Ch	31:0	reserved		

29.31.7 QUERY_HCA_VPORT_CONTEXT – Query HCA Vport Context

Table 2511 - QUERY_HCA_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
other_vport													port_num				vport_number																08h
																															0Ch		

Table 2512 - QUERY_HCA_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. Allowed only if HCA_CAP.vport_group_manager = 1.	
	19:16	port_num	Physical port number. Valid only when HCA_CAP.num_ports == 2. Otherwise reserved. 1: phy port number 1 2: phy port number 2	
	15:0	vport_number	Number of vports to be accessed within the same vports group. Valid only if other_vport = 1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	

Table 2513 - QUERY_HCA_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2513 - QUERY_HCA_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
hca_vport_context (See Table 36, “HCA_Vport Context Layout,” on page 295)																																10h-20Ch

Table 2514 - QUERY_HCA_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-20Ch	4096	hca_vport_context (See Table 36, “HCA_Vport Context Layout,” on page 295)	hca_vport context	

29.31.8MODIFY_HCA_VPORT_CONTEXT – Modify HCA Vport Context

Table 2515 - MODIFY_HCA_VPORT_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
other_vport												port_num					vport_number																08h
																																0Ch	
hca_vport_context (See Table 36, “HCA_Vport Context Layout,” on page 295)																																10h-20Ch	

Table 2516 - MODIFY_HCA_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2516 - MODIFY_HCA_VPORT_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. Allowed only if HCA_CAP.vport_group_manager =1	
	19:16	port_num	Physical port number. Valid only when HCA_- CAP.num_ports==2. Otherwise reserved. 1: phy port number 1. 2: phy port number 2.	
	15:0	vport_number	Number of vports to be accessed within the same vports group. Valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	
10h- 20Ch	4096	hca_vport_context (See Table 36, “HCA_Vport Context Lay- out,” on page 295)	hca_vport context.	

Table 2517 - MODIFY_HCA_VPORT_CONTEXT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																															00h	
syndrome																																004h
																																008h-0ch

Table 2518 - MODIFY_HCA_VPORT_CONTEXT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h- 0Ch	31:0	reserved		

29.31.9 QUERY_HCA_VPORT_GID – Query HCA VPORT GID Table

Table 2519 - QUERY_HCA_VPORT_GID Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
other_vport													port_num				vport_number																08h
																	gid_index																0Ch

Table 2520 - QUERY_HCA_VPORT_GID Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. Allowed only if HCA_CAP.vport_group_manager =1	
	19:16	port_num	Physical port number. Valid only when HCA_-CAP.num_ports==2. Otherwise reserved. 1: phy port number 1 2: phy port number 2	
	15:0	vport_number	Number of vports to be accessed within the same vports group. Valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	15:0	gid_index	GID index to read. 0xffff means read all the table.	

Table 2521 - QUERY_HCA_VPORT_GID Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
																syndrome																04h

Table 2521 - QUERY_HCA_VPORT_GID Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch
gids_num																																0Ch
gid[0]																																10h-1Ch
gid[1]																																20h-2Ch
...																																...

Table 2522 - QUERY_HCA_VPORT_GID Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
0Ch	31:16	gids_num	Number of GIDs returned.	
10h-...	128	gid[...]	List of GIDs returned. The size of the list is 1, except in the case that gid_index==0xffff, then the list size is current HCA - CAP.gid_table_size.	

29.31.10QUERY_HCA_VPORT_PKEY – Query HCA Vport Pkey Table

This command is used to query the Pkey table for IB and ROCE.

Table 2523 - QUERY_HCA_VPORT_PKEY Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
other vport												port_num					vport_number																08h
																	pkey_index																0Ch

Table 2524 - QUERY_HCA_VPORT_PKEY Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. Allowed only if HCA_CAP.vport_group_manager =1.	
	19:16	port_num	Physical port number. Valid only when HCA_CAP.num_ports==2. Otherwise reserved. 1: phy port number 1 2: phy port number 2	
	15:0	vport_number	Number of vports to be accessed within the same vports group- valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	
0Ch	15:0	pkey_index	PKEY index to read. 0xffff mean read all the table.	

Table 2525 - QUERY_HCA_VPORT_PKEY Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
																pkey[0]																10h
																pkey[1]																14h
																...																18h-...

Table 2526 - QUERY_NIC_VPORT_PKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 2526 - QUERY_NIC_VPORT_PKEY Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
10h-...	15:0	pkey[...]	List of PKeys returned. The size of the list is 1, except in case where pkey_index==0xffff, then the list size is current. HCA_CAP.pkey_table_size.	

29.32 QUERY_ROCE_ADDRESS – Query RoCE Address Entry

Table 2527 - QUERY_ROCE_ADDRESS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
roce_address_index																										vhca_port_num					08h	
																																0Ch

Table 2528 - QUERY_ROCE_ADDRESS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
~~04h~~ 04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod	Must be 0.	
08h	31:16	roce_address_index	Index in RoCE address table. Software allowed to access entries [0..HCA_CAP.roce_address_table_size-1].	
	3:0	vhca_port_num	VHCA port number.	
0Ch	31:0	reserved		

Table 2529 - QUERY_ROCE_ADDRESS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																04h				
																																08h-0Ch				
roce_address (See Table 2531, “RoCE Address Layout,” on page 2173)																																10h-2C				

Table 2530 - QUERY_ROCE_ADDRESS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-2C	256	roce_address (See Table 2531, “RoCE Address Layout,” on page 2173)	roce_address	

Table 2531 - RoCE Address Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset														
source_l3_address																																00h-0Ch														
rx allow untagged		vlan valid		vlan_id												source_mac[47:32]																10h														
																																14h														
source_mac[31:0]																																14h														
																				roce_l3_type				roce_version								18h														
																																1Ch														

Table 2532 - RoCE Address Field Descriptions

Offset	Bits	Name	Description	Access
00h-0Ch	128	source_l3_address	L3 source address. Can be 128 bits gid, 32 bits ip address or 128 bits ip address depending on roce_l3_type. Note that for IPv4 l3_type, bits[127:32] are reserved.	
10h	29	rx_allow_untagged	If set, untagged received packets will be allowed. Supported when Roce_CAP.roce_rx_allow_untagged==1. See Table 1579, “RoCE Capabilities Layout,” on page 1732.	
	28	vlan_valid	Indicates that vlan_id is valid. See Table 2533, “RoCE VLAN Behavior,” on page 2174	
	27:16	vlan_id	vlan id See Table 2533, “RoCE VLAN Behavior,” on page 2174	
	15:0	source_mac[47:32]	Upper bits of source mac address.	
14h	31:0	source_mac[31:0]	Lower bits of source mac address.	
18h	11:8	roce_l3_type	0x0: Ipv4 0x1: Ipv6 For RoCE version 1, i.e. roce_version=0, this field is reserved.	
	7:0	roce_version	0x0: version_1_0 - RoCE version 1.0 0x1: version_1_5 - RoCE version 1.5 0x2: version_2_0 - RoCE version 2.0	

Table 2533 - RoCE VLAN Behavior

vlan_valid	vlan_id	Transmit Behavior	Receive Behavior		
			Untagged Allowed	Priority Tagged Allowed	Tagged Allowed
0	NA	No vlan insertion	YES	YES	NO
1	0	Priority tagged vlan insertion	YES	YES	NO
1	!= 0	vlan insertion	NO	NO	YES

29.33 SET_ROCE_ADDRESS – SET RoCE Address Entry

Table 2534 - SET_ROCE_ADDRESS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
vhca_tunnel_id																op_mod																04h
roce_address_index																										vhca_port_num				08h		
																																0Ch
roce_address (See Table 2531, “RoCE Address Layout,” on page 2173)																																10h-2Ch

Table 2535 - SET_ROCE_ADDRESS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	31:16	vhca_tunnel_id	When not 0, indicates this commands is set on behalf on another vhca and this field indicates the identifier returned when creating VHCA_TUNNEL_OBJECT.	
	15:0	op_mod		
08h	31:16	roce_address_index	Index in RoCE address table. Software allowed to access entries [0..HCA_CAP.roce_address_table_size-1]	
	3:0	vhca_port_num	VHCA port number.	
10h-2Ch	256	roce_address (See Table 2531, “RoCE Address Layout,” on page 2173)	Roce_address.	

Table 2536 - SET_ROCE_ADDRESS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 2536 - SET_ROCE_ADDRESS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 008h-0Ch

Table 2537 - SET_ROCE_ADDRESS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	31:0	reserved		

29.34 Vport ~~Counters~~Monitoring Commands

Table 2538 - Vport monitoring Commands Overview

Command	Description	Reference
QUE- RY_VPORT_ COUNTER	Query vport traffic counters	Section 29.34.1, “QUE- RY_VPORT_COUNTER – Query Vport Counter,” on page 2176
QUE- RY_ ~~VPORT~~ - VNIC_ENV	Query NIC/HCA Vport statistics, sta- tus and monitors	

29.34.1QUERY_VPORT_COUNTER – Query Vport Counter

When a Vport is created, a set of traffic counters on this Vport will also be automatically created.

Vport counters are supported only when HCA_CAP.vport_counters ==1.

Table 2539 specifies the traffic counters for each Vport.

Table 2539 - Vport Counters List

Counter Name	Description
Received errors	Number of received error packets/octets
Transmit errors	Number of transmit error packets/octets

Table 2539 - Vport Counters List

Counter Name	Description
Received IB unicast	Number of received IB unicast packets/octets
Transmitted IB unicast	Number of transmitted IB unicast packets/octets
Received IB multicast	Number of received IB multicast packets/octets
Transmitted IB multicast	Number of transmitted IB multicast packets/octets
Received ETH broadcast	Number of received Ethernet broadcast packets/octets
Transmitted broadcast	Number of transmitted Ethernet broadcast packets/octets
Received ETH unicast	Number of received Ethernet unicast packets/octets
Transmitted ETH unicast	Number of transmitted Ethernet unicast packets/octets
Received ETH multicast	Number of received Ethernet multicast packets/octets
Transmitted ETH multicast	Number of transmitted Ethernet multicast packets/octets

29.34.2 QUERY_VPORT_COUNTER – Query Vport Counter

QUERY_VPORT_COUNTER command is used to query and reset traffic counters on a specific Vport.

Table 2540 - QUERY_VPORT_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other_vport												port_num		vport_number																08h		
																															0Ch-14h	

Table 2540 - QUERY_VPORT_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
clear																																18h
																																1Ch

Table 2541 - QUERY_VPORT_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: vport_counters others: reserved	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. allowed only if HCA_CAP.vport_group_manager =1	
	19:16	port_num	Physical port number.valid only when HCA - CAP.num_ports==2, otherwise reserved. 1: phy port number 1. 2: phy port number 2.	
	15:0	vport_number	Number of vports to be accessed within the same vports group- valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	
18h	31	clear	Read and reset counter value	

Table 2542 - QUERY_VPORT_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
received_errors (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																10h-1Ch
transmit_errors (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																20h-2Ch

Table 2542 - QUERY_VPORT_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
received_ib_unicast(See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																30h-3Ch
transmitted_ib_unicast(See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																40h-4Ch
received_ib_multicast(See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																50h-5Ch
transmitted_ib_multicast(See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																60h-6Ch
received_eth_broadcast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																70h-7Ch
transmitted_eth_broadcast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																80h-8Ch
received_eth_unicast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																90h-9Ch
transmitted_eth_unicast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																A0h-ACh
received_eth_multicast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																B0h-BCh
transmitted_eth_multicast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)																																C0h-CCh
																																D0h-20Ch

Table 2543 - QUERY_VPORT_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2543 - QUERY_VPORT_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h-1Ch	128	received_errors (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received error packets/octets.	
20h-2Ch	128	transmit_errors (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of transmit error packets/octets.	
30h-3Ch	128	received_ib_unicast(See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received ib unicast packets/octets.	
40h-4Ch	128	transmitted_ib_unicast(See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of transmitted ib unicast packets/octets.	
50h-5Ch	128	received_ib_multicast(See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received ib multicast packets/octets.	
60h-6Ch	128	transmitted_ib_multi-cast(See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)	Number of transmitted ib multicast packets/octets.	
70h-7Ch	128	received_eth_broadcast (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received broadcast packets/octets.	
80h-8Ch	128	transmitted_eth_broad-cast (See Table 2544, “TRAFFIC_COUNTER Format Layout,” on page 2181)	Number of transmitted broadcast packets/octets.	
90h-9Ch	128	received_eth_unicast (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received unicast Ethernet packets/octets.	
A0h-ACh	128	transmitted_eth_unicast (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of transmitted unicast Ethernet packets/octets.	

Table 2543 - QUERY_VPORT_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
B0h-BCh	128	received_eth_multicast (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of received multicast Ethernet packets/octets.	
C0h-CCh	128	transmitted_eth_multicast (See Table 2544, “TRAF-FIC_COUNTER Format Layout,” on page 2181)	Number of transmitted multicast Ethernet packets/octets.	

Table 2544 - TRAFFIC_COUNTER Format Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
packets[63:32]																															00h	
packets[31:0]																															04h	
octets[63:32]																															08h	
octets[31:0]																															0Ch	

Table 2545 - TRAFFIC_COUNTER Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	packets[63:32]	Upper bits of total number of packets.	
04h	31:0	packets[31:0]	Lower bits of total number of packets.	
08h	31:0	octets[63:32]	Upper bits of total number of octets.	
0Ch	31:0	octets[31:0]	Lower bits of total number of octets.	

29.34.3QUERY_VNIC_ENV – Query vNIC Environment

Supported only for vport_group_manager for other functions that aren't vport_group_managers, except of ECPF that can perform it from all functions.

[Internal] To not break compatibility, device will return 0 and not fail command for not allowed access.

Table 2546 - QUERY_VNIC_ENV Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
other vport																vport_number																08h

Table 2547 - QUERY_VNIC_ENV Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	0: vport_diag_statistics others: reserved	
08h	31	other_vport	0 - access my vport. vport_number is reserved. 1 - access another vport determined by vport_number. allowed only if HCA_CAP.vport_group_manager =1	
	15:0	vport_number	Number of vports to be accessed within the same vports group- valid only if other_vport =1. For more details, See Section 14.1.2, “Virtual Ports,” on page 839.	

Table 2548 - QUERY_VNIC_ENV Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
vportnic_env																																10h-20Ch

Table 2549 - QUERY_VNIC_ENV Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-20Ch	4096	vnic_env	For op_mod vnic_diag_statistics	

Table 2550 - vNIC Diagnostic Statistics Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
current_q_under_processor_handle																															00h	
total_q_under_processor_handle																															04h	
send_queue_priority_update_flow																															08h	
																															0Ch	
nic_receive_steering_discard																															10h-14h	
receive_discard_vport_down																															18h-1Ch	
transmit_discard_vport_down																															20h-24h	
async_eq_overrun																															28h	
comp_eq_overrun																															2C	
																								nic_receive_steering_depth							30h	
invalid_command																															34h	
quota_exceeded_command																															38h	
internal_rq_out_of_buffer																															3Ch	
																															40h-1FCh	

Table 2551 - vNIC Diagnostic Statistics Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	current_q_under_processor_handle	The current number of qs that are handled as exception flow due to async error or to errored cmd Supported only when HCA_CAP.vnic_env_queue_counters==1.	
04h	31:0	total_q_under_processor_handle	The total number of qs that are handled as exception flow due to async error or to errored cmd. Supported only when HCA_CAP.vnic_env_queue_counters==1.	
08h	31:0	send_queue_priority_update_flow	the total number of QP/SQ priority/SL update events. Supported only when HCA_CAP.vnic_env_queue_counters==1.	
10h-14h	64	nic_receive_steering_discard	Number of packets that completed the NIC Receive Flow Table steering, and discarded because they did not match any Flow in the final Flow Table. Supported only when HCA_CAP.nic_receive_steering_discard==1.	
18h-1Ch	64	receive_discard_vport_down	Number of packets that were steered to a VPort, and discarded because the VPort was not in a state to receive packets. Supported only when HCA_CAP.receive_discard_vport_down==1.receive_discard_vport_down==1.	
20h-24h	64	transmit_discard_vport_down	Number of packets that were transmitted by a vNIC, and discarded because the VPort was not in a state to transmit packets. Supported only when HCA_CAP.transmit_discard_vport_down==1.	
28h	31:0	async_eq_ouerrun	Number of times an EQ mapped to Async. events was overrun. Note: the counter cannot be assumed to be reset upon driver initialization. Supported only when HCA_CAP.eq_ouerrun_count==1.	
2Ch	31:0	comp_eq_ouerrun	Number of times an EQ mapped to Completion events was overrun. Note: the counter cannot be assumed to be reset upon driver initialization. Supported only when HCA_CAP.eq_ouerrun_count==1.	
30h	7:0	nic_receive_steering_depth	The current evaluation of the longest possible steering path a packet may traverse through the NIC receive steering table. The field is given in device specific units. Supported only when HCA_CAP.nic_receive_steering_depth==1. Note: value is undefined if a limit is not set by SET_HCA_CAP	

Table 2551 - vNIC Diagnostic Statistics Field Descriptions

Offset	Bits	Name	Description	Access
34h	31:0	invalid_command	Number commands issued by the VF and failed for any reason other than quota exceeded. Supported when HCA_CAP.invalid_command_count==1.	
38h	31:0	quota_exceeded_command	Number of commands issued by the VF and failed due to quota exceeded. Supported when HCA_CAP.quota_exceeded_count==1.	
3Ch	31:0	internal_rq_out_of_buffer	Number of dropped packets due to lack of receive WQEs for an internal device RQs. Supported only when HCA_CAP.vnic_env_int_rq_oob==1. [Internal]: Hairpin RQ for both rx and tx direction that used to implement WAS like: TTL, VGT+, mirroring of vf.	

29.35 Q Counters Commands

Q counters can be associated with one or more QPs to collect statistics on the monitored QP.

Each Q counter set includes a number of counters as described in Table 2563, “QUERY_Q_COUNTER Output Structure Field Descriptions”.

Before associating a q_counter_set to a QP, a Q counter set must be allocated using ALLOC_Q_COUNTER.

The ALLOC_Q_COUNTER returns counter_set_id which can be used as a handle for attaching the Q counter set to QP, query of the Q counter set and deallocating of the Q counter set.

Upon allocation, all counters within the counter set are set to 0

DEALLOC_Q_COUNTER deallocates a q_counter_set. Deallocating a Q counter set which has a QP associated to it will fail. A counter which was deallocated cannot be read.

QUERY_Q_COUNTER command is used to query and reset a counter set specified by the counter_set_id.

The max number of Q_counters is defined in HCA_CAP.max_qp_cnt.

29.35.1ALLOC_Q_COUNTER - Allocate a Q-Counter

ALLOCATE_Q_COUNTER command is used to allocate a new Q counter set and return its counter_set_id.

Table 2552 - ALLOC_Q_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				

Table 2552 - ALLOC_Q_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																op_mod												04h				
																																08h-0Ch

Table 2553 - ALLOC_Q_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539 uid=0xffff indicates this resource is shared and can be used by any User.	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 2554 - ALLOC_Q_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																								counter_set_id								08h
																																0Ch

Table 2555 - ALLOC_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	7:0	counter_set_id	Q counter Set ID	
0Ch	31:0	reserved		

29.35.2 DEALLOC_Q_COUNTER - De-Allocate Q-Counter

Table 2556 - DEALLOC_Q_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																							counter_set_id						08h			
																																0Ch

Table 2557 - DEALLOC_Q_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	7:0	counter_set_id	Q counter Set ID	
0Ch	31:0	reserved		

Table 2558 - DEALLOC_Q_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2559 - DEALLOC_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.35.3 QUERY_Q_COUNTER – Query Q Counter

QUERY_Q_COUNTER command is used to query and reset a counter set specified by the counter_set_id.

Table 2560 - QUERY_Q_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-14h
clear																																18h
																								counter_set_id								1Ch

Table 2561 - QUERY_Q_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Must be 0	
18h	31	clear	read and reset counter value. when set to 1, HW will reset counters.	
1Ch	7:0	counter_set_id	Counter set identifier to read/read and clear.	

Table 2562 - QUERY_Q_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch
rx_write_requests																																10h
																																14h
rx_read_requests																																18h
																																1Ch

Table 2562 - QUERY_Q_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rx_atomic_requests																																20h
																																24h
rx_dct_connect																																28h
																																2Ch
out_of_buffer																																30h
																																34h
out_of_sequence																																38h
																																3Ch
duplicate_request																																40h
																																44h
rnr_nak_retry_err																																48h
																																4Ch
packet_seq_err																																50h
																																54h
implied_nak_seq_err																																58h
																																5Ch
local_ack_timeout_err																																60h
																																64h
resp_rnr_nak																																68h
																																6Ch
req_rnr_retries_exceeded																																70h
																																74h
resp_local_length_error																																78h
req_local_length_error																																7Ch
resp_local_qp_error																																80h
local_operation_error																																84h
resp_local_protection																																88h
req_local_protection																																8Ch

Table 2562 - QUERY_Q_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
resp_cqe_error																																90h
req_cqe_error																																94h
req_memory_window_binding																																98h
req_bad_response																																9Ch
req_remote_invalid_request																																A0h
resp_remote_invalid_request																																A4h
req_remote_access_errors																																A8h
resp_remote_access_errors																																ACH
req_remote_operation_errors																																B0h
req_transport_retries_exceeded																																B4h
cq_overflow																																B8h
resp_cqe_flush_error																																BCh
req_cqe_flush_error																																C0h
max_qp_retry_freq_exceeded																																C4h
roce_adp_retrans																																C8h
roce_adp_retrans_to																																CCh
roce_slow_restart																																D0h
roce_slow_restart_cnps																																D4h
roce_slow_restart_trans																																D8h
																																DCh-FCh

Table 2563 - QUERY_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2563 - QUERY_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	rx_write_requests	Number of received WRITE request for the associated QPs <u>TBD- add CAP that this counter supported</u> Supported for IB only.	
18h	31:0	rx_read_requests	Number of received READ request for the associated QPs Supported for IB only.	
20h	31:0	rx_atomic_requests	Number of received ATOMIC request for the associated QPs. Supported for IB only.	
28h	31:0	rx_dct_connect	Number of received connection request for the associated DCTs. Supported for IB only.	
30h	31:0	out_of_buffer	Number of dropped occurred due to lack of WQE for the associated QPs/RQs.	
38h	31:0	out_of_sequence	Supported when HCA_CAP.out_of_seq_cnt==1 and only for RC, UC, UD, XRC, DCT transport services. See Table 1571, "HCA Capabilities Layout," on page 1671. [Internal - shomron only]	
40h	31:0	duplicate_request	Number of received duplicate request packets. A duplicate request is a request that has been previously executed. Supported when HCA_CAP.retransmission_q_counters==1 and only for XRC, DCT and RC transport services.	
48h	31:0	rn timer_retry_err	Number of received RNR NAK packets. The QP retry limit was not exceeded. Supported when HCA_CAP.retransmission_q_counters==1 and only for XRC,DCT and RC transport services.	
50h	31:0	packet_seq_err	Number of received NAK-Sequence error packets. The QP retry limit was not exceeded. Supported when HCA_CAP.retransmission_q_counters==1 and only for XRC,DCT and RC transport services.	
58h	31:0	implied_nak_seq_err	Number of times the Requester detected an ACK with a PSN larger than the expected PSN for an RDMA READ or ATOMIC response. The QP retry limit was not exceeded. Supported when HCA_CAP.retransmission_q_counters==1 and only for XRC,DCT and RC transport services.	

Table 2563 - QUERY_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
60h	31:0	local_ack_timeout_err	Number of No ACK response from responder within timer interval. The QP retry limit was not exceeded. Supported when HCA_CAP.retransmission_q_counters==1 and only for XRC,DCT and RC transport services.	
68h	31:0	resp_rnr_nak	Number of times the responder sent RNR (receiver Not Ready) NAKs. Supported when HCA_CAP.rnr_nak_q_counters ==1 and only for DCT and RC transport services.	
70h	31:0	req_rnr_retries_exceeded	requester_rnr_retries_exceeded – number of times the requester RNR NAKs retries exceeded errors. Supported when HCA_CAP.rnr_nak_q_counters ==1 and only for DCT and RC transport services.	
78h	31:0	resp_local_length_error	Number of times responder detected local length errors. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services	
7Ch	31:0	req_local_length_error	Number of times requester detected local length errors. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and for RC, UC, UD, XRC, DCT and Eth transport services	
80h	31:0	resp_local_qp_error	Number of times responder detected local QP operation errors. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services	
84h	31:0	local_operation_error	Number of times requester detected local QP operation errors. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services	
88h	31:0	resp_local_protection	Number of times responder detected memory protection error in its local memory subsystem. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services	
8Ch	31:0	req_local_protection	Number of times requester detected memory protection error in its local memory subsystem. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services.	
90h	31:0	resp_cqe_error	Number of CQEs completed with error in responder. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, DCT and Eth transport services	

Table 2563 - QUERY_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
94h	31:0	req_cqe_error	Number of CQEs completed with error in requester. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, UC, UD, XRC, and Eth transport services.	
98h	31:0	req_memory_window_binding	Number of times requester detected memory window binding error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and for RC, UC, UD, XRC, DCT and Eth transport services.	
9Ch	31:0	req_bad_response	Number of times requester detected bad responses. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport services.	
A0h	31:0	req_remote_invalid_request	Number of times requester detected remote invalid request error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
A4h	31:0	resp_remote_invalid_request	Number of times responder detected remote invalid request error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
A8h	31:0	req_remote_access_errors	Number of times requester detected remote access error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport	
ACh	31:0	resp_remote_access_errors	Number of times responder detected remote access error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC, UC and DCT transport	
B0h	31:0	req_remote_operation_errors	Number of times requester detected remote operation error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
B4h	31:0	req_transport_retries_exceeded	Number of times requester detected transport retries exceed error. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
B8h	31:0	cq_overflow	Number of times this Q tried to generate a CQE and failed due to full CQ. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport	

Table 2563 - QUERY_Q_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
BCh	31:0	resp_cqe_flush_error	Number of CQEs completed with flushed with error in responder. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
C0h	31:0	req_cqe_flush_error	Number of CQEs completed with flushed with error in requester. Supported when HCA_CAP.enhanced_retransmission_q_counters==1 and only for RC, XRC and DCT transport.	
C4h	31:0	max_qp_retry_freq_exceeded	Incremented each time a QP is moved to error state due to QP.max_qp_retry_limit/period (see max_qp_retry_period in Table 29, “NIC_Vport Context Layout,” on page 287).	
C8h	31:0	roce_adp_retrans	Number of adaptive retransmission for RoCE traffic. Supported when HCA_CAP.roce_accl=1. Only for RoCE traffic (RC, UD)	
CCh	31:0	roce_adp_retrans_to	Number of times RoCE traffic reached timeout due to adaptive retransmission. Supported when HCA_CAP.roce_accl=1. Only for RoCE traffic (RC, UD)	
D0h	31:0	roce_slow_restart	Number of times RoCE slow restart was used. Supported when HCA_CAP.roce_accl=1. Only for RoCE traffic (RC, UD)	
D4h	31:0	roce_slow_restart_cnps	Number of times RoCE slow restart generate CNP packets, Supported when HCA_CAP.roce_accl=1. Only for RoCE traffic (RC, UD)	
D8h	31:0	roce_slow_restart_trans	Number of times RoCE slow restart change state to slow restart. Supported when HCA_CAP.roce_accl=1. Only for RoCE traffic (RC, UD)	

29.35.4 SET_MONITOR_COUNTER - Set Monitor Counter

Table 2564 - SET_MONITOR_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																num_of_counters																08h
																																0Ch
monitor_counter[0](See Table 2569, “MONITOR_COUNTER Output Structure Field Descriptions,” on page 2196)																																10h-14h
monitor_counter[1]																																18h-1Ch
...																																20h-..

Table 2565 - SET_MONITOR_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod		
08h	15:0	num_of_counters	Number of counters to monitor.	
10h-..	64	monitor_counter[...] (See Table 2569, "MONITOR_COUNTER Output Structure Field Descriptions," on page 2196)	List of counter to monitor. See Table 2569, "MONITOR_COUNTER Output Structure Field Descriptions," on page 2196.	

Table 2566 - SET_MONITOR_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																	00h
syndrome																																004h	

Table 2566 - SET_MONITOR_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 2567 - SET_MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

Table 2568 - MONITOR_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				type												counter																00h
counter_group_id																																04h

Table 2569 - MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	27:24	type	Counter Type 0x0: PPCNT_COUNTERS - Support only when HCA_CAP.num_ppcnt_monitor_counters>0 which specify number of counters supported in counter id. 0x1: Q_COUNTER - Support only when HCA_CAP.num_q_monitor_counters>0 which specify number of counters supported in counter id.	

Table 2569 - MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	counter	Counter identifier according to type and group. if type ==PPCNT_COUNTERS (The supported counter are according to HCA_CAP.num_ppcnt_monitor_counters): 0x0: ppcent.802_3.a_in_range_length_errors 0x1: ppcent.802_3.a_out_of_range_length_field 0x2: ppcent.802_3.a_frame_too_long_errors 0x3: ppcent.802_3.a_frame_check_sequence_errors 0x4: ppcent.802_3.a_alignment_errors 0x5: ppcent.802_3.if_out_discards If if type ==Q_COUNTER (The supported counter are according to HCA_CAP.num_q_monitor_counters): 0x0: rx_write_requests 0x1: rx_read_requests 0x2: rx_atomic_requests 0x3: rx_dct_connect 0x4: out_of_buffer 0x5: out_of_sequence 0x6: duplicate_request 0x7: rnr_nak_retry_err 0x8: packet_seq_err 0x9: implied_nak_seq_err 0xa: local_ack_timeout_err 0xb: resp_rnr_nak 0xc: req_rnr_retries_exceeded 0xd: resp_local_length_error 0xe: req_local_length_error 0xf: resp_local_qp_error 0x10: local_operation_error 0x11: resp_local_protection 0x12: req_local_protection 0x13: resp_cqe_error 0x14: req_cqe_error 0x15: req_memory_window_binding 0x16: req_bad_response 0x17: req_remote_invalid_request 0x18: resp_remote_invalid_request 0x19: req_remote_access_errors 0x1a: resp_remote_access_errors 0x1b: req_remote_operation_errors 0x1c: req_transport_retries_exceeded 0x1d: cq_overflow 0x1e: resp_cqe_flush_error 0x1f: req_cqe_flush_error 0x20: max_qp_retry_freq_exceeded	

Table 2569 - MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	counter_group_id	Indicates the group of counters according to type. if type == PPCNT_COUNTERS, then reserved and counters only indicated by counter_id. if type == Q_COUNTER, then indicates q_counter_set and counter_id specifies the counter.	

29.35.5 ARM_MONITOR_COUNTER - ARM Monitor Counter

The device can monitor specific counters and provide an event to notify when these counters are changed.

The monitoring is done in a best effort manner where the minimum notification period is 200 ms. However, when the device is loaded, the notification may be delayed.

To configure the required counters to be monitored, the SET_MONITOR_COUNTER command shall be used with a list of counters to be monitored.

The monitor counter can monitor up to HCA_CAP.max_num_of_monitor_counters.

The configuration is done based on counter type (such as ppcnt, q counter, etc.), and additional parameters according to the type of counter selected.

Upon monitored counter change, the device will generate a Monitor_Counter_Change event.

In order to avoid loss of events, the device will not generate any new event before software re-arms the monitoring functionality, using ARM_MONITOR_COUNTER command.

Note: Setting the monitored counters does not arm events. User shall call ARM_COUNTERS_MONITOR in order to arm.

Table 2570 - ARM_MONITOR_COUNTER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2571 - ARM_MONITOR_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	

Table 2571 - ARM_MONITOR_COUNTER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		

Table 2572 - ARM_MONITOR_COUNTER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																																004h
																																08h-0Ch

Table 2573 - ARM_MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.36 QoS Commands - Quality of Service Commands

29.36.1SET_PP_RATE_LIMIT – Set Packet Pacing Rate Limit

The SET_PP_RATE_LIMIT command configures an entry in the Rate Limit table for Packet Pacing. Supported when QoS_CAP.packet_pacing== 1.

Table 2574 - SET_PP_RATE_LIMIT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
																rate_limit_index												08h				
																																0Ch

Table 2574 - SET_PP_RATE_LIMIT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
pp_context (Table 2578, “PP_CONTEXT Input Structure Layout,” on page 2201)																																180h-3Ch

Table 2575 - SET_PP_RATE_LIMIT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	15:0	rate_limit_index	Index inside the Rate Limit table. 0x0 is reserved	
10h-3Ch	384	pp_context (Table 2578, “PP_CONTEXT Input Structure Layout,” on page 2201)	Packet Pacing context	

Table 2576 - SET_PP_RATE_LIMIT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																												004h				
																												08h-0Ch				

Table 2577 - SET_PP_RATE_LIMIT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

Table 2578 - PP_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rate_limit																															00h	
burst_upper_bound																															04h	
															typical_packet_size																08h	
																															0Ch-2Ch	

Table 2579 - PP_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	rate_limit	Maximal transmission rate, given in Kb/s	
04h	31:0	burst_upper_bound	Upper bound for packet burst, given in bytes. The device will schedule bursts of packets for an SQ connected to this rate, smaller than or equal to this value. Value 0x0 indicates packet bursts will be limited to the device defaults. This field should be used if bursts of packets must be strictly kept under a certain value. Supported only when HCA_CAP.QoS.packet_pacing_burst_bound==1.	
08h	15:0	typical_packet_size	The expected size for packets associated with this rate limit, given in bytes. Value 0x0 indicates the packet size is unknown, and assumed to be MTU. Supported only when HCA_CAP.QoS.packet_pacing_typical_size==1.	

29.36.2QUERY_PP_RATE_LIMIT – Query Packet Pacing Rate Limit

The QUERY_PP_RATE_LIMIT command reads configured entry in the Rate Limit table for Packet Pacing.Supported when QoS_CAP.packet_pacing== 1.

Table 2580 - QUERY_PP_RATE_LIMIT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																rate_limit_index																08h
																																0Ch

Table 2581 - QUERY_PP_RATE_LIMIT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	15:0	rate_limit_index	Index inside the Rate Limit table. 0x0 is reserved	

Table 2582 - QUERY_PP_RATE_LIMIT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch
pp_context (Table 2578, “PP_CONTEXT Input Structure Layout,” on page 2201)																																180h-3Ch

Table 2583 - QUERY_PP_RATE_LIMIT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-3Ch	384	pp_context (Table 2578, “PP_CONTEXT Input Structure Layout,” on page 2201)	Packet Pacing context	

29.36.3CREATE_SCHEDULING_ELEMENT - Create a Transmit Scheduling element

This commands creates a new Transmit Scheduling element:

- TSAR
- VPort
- VPort-TC pair

- **Para-VPort-TC pair**

When completed successfully, a new element is created, attached to an existing element (except for the root TSAR). See “TSAR Configuration” on page 538.

Table 2584 - CREATE_SCHEDULING_ELEMENT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
scheduling_hierarchy																																08h
																																10h-1Ch
scheduling_context (See Table 2586, “Scheduling Context Format,” on page 2204)																																20h-5Ch
																																60h-7Ch

Table 2585 - CREATE_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:24	scheduling_hierarchy	Location of the element in Scheduling hierarchy 0x2: E_SWITCH 0x3: NIC other values are reserved	
20h-5Ch	512	scheduling_context (See Table 2586, “Scheduling Context Format,” on page 2204)	Scheduling element context.	

Table 2586 - Scheduling Context Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
element_type																																00h
element_attributes																																04h
parent_element_id																																08h
																																10h
bw_share																																14h
max_average_bw																																18h
																																1Ch-3Ch

Table 2587 - Scheduling Context Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	element_type	Type of Scheduling Element 0x0: TSAR 0x1: VPORT 0x2: VPORT_TC 0x3: PARA_VPORT_TC 0x4: QUEUE_GROUP other values are reserved	
04h	31:0	element_attributes	Specific attributes according to the <i>element_type</i> . TSAR - See Table 2588, “TSAR Element Attributes Format,” on page 2205 VPORT - See Table 2590, “VPORT Element Attributes Format,” on page 2205 VPORT_TC - See Table 2592, “VPORT_TC Element Attributes Format,” on page 2206 PARA_VPORT_TC - See Table 2594, “PARA_VPORT_TC Element Attributes Format,” on page 2206	
08h	31:0	parent_element_id	Identifier of the Scheduling element to which the created Scheduling element will be connected. This field is reserved for the root TSAR.	
14h	31:0	bw_share	The relative BW share allocated for this element. This field has no units. The BW is shared between all elements connected to the same parent element, relatively to their <i>bw_share</i> . Value 0x0 indicates a device default Weight. This field is reserved for the root TSAR	

Table 2587 - Scheduling Context Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:0	max_aver- age_bw	The maximal transmission rate allowed for the element, averaged over time. Value is given in units of 1 Mbit/sec. Value 0x0 indicates the rate is unlimited.	

Table 2588 - TSAR Element Attributes Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
traffic_class				tsar_type																							00h					

Table 2589 - TSAR Element Attributes Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 8	traffic_class	Traffic Class. This TSAR represents tclass when connected to an ETS TSAR. If the TSAR's parent is not an ETS TSAR, this field is reserved.	
	23:1 6	tsar_type	Type of TSAR 0x0: DWRR 0x1: ROUND_ROBIN 0x2: ETS other values are reserved	

Table 2590 - VPORT Element Attributes Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																vport_number												00h				

Table 2591 - VPORT Element Attributes Field Descriptions

Offset	Bits	Name	Description	Access
00h	15:0	vport_number		

Table 2592 - VPORT_TC Element Attributes Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
traffic_class								vport_number																								00h

Table 2593 - VPORT_TC Element Attributes Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	traffic_class		
	15:0	vport_number		

Table 2594 - PARA_VPORT_TC Element Attributes Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
traffic_class								qos_para_vport_number																								00h

Table 2595 - PARA_VPORT_TC Element Attributes Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:28	traffic_class		
	15:0	qos_para_vport_number		

Table 2596 - CREATE_SCHEDULING_ELEMENT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
																syndrome																04h
																																08h
																scheduling_element_id																10h
																																14h-3Ch

Table 2597 - CREATE_SCHEDULING_ELEMENT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		
10h	31:0	scheduling_element_id	Identifier for the created Scheduling element	

29.36.4 DESTROY_SCHEDULING_ELEMENT - Destroy a Transmit Scheduling Element

This commands destroys an existing Transmit Scheduling element. When completed successfully, all resources associated with the element are freed. An element may be destroyed only if no agents are connected to it. See “TSAR Configuration” on page 538.

Table 2598 - DESTROY_SCHEDULING_ELEMENT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid												00h				
																op_mod												04h				
scheduling_hierarchy																											08h					
scheduling_element_id																																0Ch
																																10h-3Ch

Table 2599 - DESTROY_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:1 6	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:2 4	scheduling_hierarchy	Location of the element in Scheduling hierarchy 0x2: E_SWITCH 0x3: NIC other values are reserved	

Table 2599 - DESTROY_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31:0	scheduling_element_id	Identifier for the destroyed Scheduling element	

Table 2600 - DESTROY_SCHEDULING_ELEMENT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																												04h				
																												08h-3Ch				

Table 2601 - DESTROY_SCHEDULING_ELEMENT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.36.5 QUERY_SCHEDULING_ELEMENT - Query a Transmit Scheduling Element

This commands queries an existing Transmit Scheduling element. See “TSAR Configuration” on page 538.

Table 2602 - QUERY_SCHEDULING_ELEMENT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
scheduling_hierarchy																																08h
scheduling_element_id																														0Ch		
																														10h-3Ch		

Table 2603 - QUERY_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:24	scheduling_hierarchy	Location of the element in Scheduling hierarchy 0x2: E_SWITCH 0x3: NIC other values are reserved	
0Ch	31:0	scheduling_element_id	Identifier for the queried Scheduling element	

Table 2604 - QUERY_SCHEDULING_ELEMENT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-1Ch
scheduling_context (See Table 2586, “Scheduling Context Format,” on page 2204)																																20h-5Ch
																																60h-7Ch

Table 2605 - QUERY_SCHEDULING_ELEMENT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2605 - QUERY_SCHEDULING_ELEMENT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
20h-5Ch	512	scheduling_context (See Table 2586, “Scheduling Context Format,” on page 2204)	Scheduling element context.	

29.36.6MODIFY_SCHEDULING_ELEMENT - Modify a Transmit Scheduling Element Attributes

This command modifies the QoS attributes assigned to Scheduling element. See “TSAR Configuration” on page 538.

Table 2606 - MODIFY_SCHEDULING_ELEMENT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
scheduling_hierarchy																																08h
scheduling_element_id																																0Ch
																																10h
modify_bitmask																																14h
																																18-1Ch
scheduling_context (See Table 2586, “Scheduling Context Format,” on page 2204)																																20h-5Ch
																																60h-7Ch

Table 2607 - MODIFY_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2607 - MODIFY_SCHEDULING_ELEMENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	31:24	scheduling_hierarchy	Location of the element in Scheduling hierarchy 0x2: E_SWITCH other values are reserved	
0Ch	31:0	scheduling_element_id	Identifier for the modified Scheduling element	
14h	31:0	modify_bitmask	Bit per Scheduling element attribute. When set, the appropriate attribute will be modified. When cleared, the appropriate attribute will remain unmodified. Bit 0: <i>bw_share</i> Bit 1: <i>max_average_bw</i> Other bits are reserved.	
20h-5Ch	512	scheduling_context (See Table 2586, "Scheduling Context Format," on page 2204)	Scheduling element context. Only fields requested by the mask should be valid. Other fields are reserved.	

Table 2608 - MODIFY_SCHEDULING_ELEMENT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-3Ch

Table 2609 - MODIFY_SCHEDULING_ELEMENT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.36.7 CREATE_QOS_PARA_VPORT - Create an E-Switch Scheduler Para-VPort

This command creates a new Para-VPort for the E-Switch. See “Para-VPort” on page 925.

Table 2610 - CREATE_QOS_PARA_VPORT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-3Ch

Table 2611 - CREATE_QOS_PARA_VPORT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2612 - CREATE_QOS_PARA_VPORT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																																04h
																																08h
																qos_para_vport_number																0Ch
																																10h-3Ch

Table 2613 - CREATE_QOS_PARA_VPORT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		

Table 2613 - CREATE_QOS_PARA_VPORT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	31:0	syndrome		
0Ch	15:0	qos_para_vport_number	Identifier for the created Para-Vport	

29.36.8 DESTROY_QOS_PARA_VPORT - Destroy an E-Switch SchedulerPara-VPort

This command destroys an existing Para-VPort for the E-Switch. See “Para-VPort” on page 925.

Table 2614 - DESTROY_QOS_PARA_VPORT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																qos_para_vport_number																0Ch
																																10h-3Ch

Table 2615 - DESTROY_QOS_PARA_VPORT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	15:0	qos_para_vport_number	Identifier for the destroyed Para-Vport	

Table 2616 - DESTROY_QOS_PARA_VPORT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h

Table 2616 - DESTROY_QOS_PARA_VPORT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
syndrome																																04h
																																08h-3Ch

Table 2617 - DESTROY_QOS_PARA_VPORT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:2 4	status		
04h	31:0	syndrome		

29.37 Multicast Groups Commands

In the following text, unless otherwise noted, “Multicast Group” refers to any Steering Group, as the steering as well as the multicast, can replicate packets to one or more destinations.

Multicast and Steering Group commands are used for managing the multicast and steering tables in the hardware to control multicast steering.

The following commands are used to set up and maintain multicast groups.

Table 2618 - Multicast and Steering Commands

Mnemonic	Command	Description	Reference
ATTACH_TO_MCG	Attach QP to multicast group	Attaches a QP to multicast group	Section 29.37.1, on page 2214
DETACH_FROM_MCG	Detach QP from multicast group	Detaches a QP from multicast group	Section 29.37.2, on page 2216

29.37.1 ATTACH_TO_MCG - Attach a QP to Multicast Group

The ATTACH_TO_MCG command attaches a QP to multicast group. If a multicast group does not exist, the device will automatically create it.

Table 2619 - ATTACH_TO_MCG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h

Table 2619 - ATTACH_TO_MCG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																op_mod																04h
								qpn																				08h				
qpn_vhca_id valid																qpn_vhca_id																0Ch
multicast_gid																																10-1Ch

Table 2620 - ATTACH_TO_MCG Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	qpn	QP number	
0Ch	31	qpn_vhca_id_valid	If set, the qpn_vhca_id field is valid Supported only when HCA_CAP.m_pci_port==1. See Section 14.9, “Multi PCI Port,” on page 857	
	15:0	qpn_vhca_id	Valid only when qpn_vhca_id_valid==1. Specifies the vHCA of the QP specified by the qpn field. See Section 14.9, “Multi PCI Port,” on page 857	
10h-1Ch	128	multicast_gid		

Table 2621 - ATTACH_TO_MCG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																00-4h

Table 2621 - ATTACH_TO_MCG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 2622 - ATTACH_TO_MCG Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.37.2 DETACH_FROM_MCG - Detach a QP from multicast group

The DETACH_FROM_MCG command detaches a QP from multicast group. If the QP is the last QP in the multicast group, the device will automatically close it.

Table 2623 - DETACH_FROM_MCG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																qpn																08h
																																0Ch
multicast_gid																																10-1Ch

Table 2624 - DETACH_FROM_MCG Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	23:0	qpn	QP number	
10h-1Ch	128	multicast_gid		

Table 2625 - DETACH_FROM_MCG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2626 - DETACH_FROM_MCG Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.38 IB Port sniffer

[Internal]- Note - We added it due to MS request for IB port sniffing as no time was to add flow table sniffer for IB.

The ATTACH_TO_SNIFFER command attaches a send/receive port traffic to a sniffer QP. The sniffer QP is a UD QP created prior to executing attach command. There can be a single Sniffer QP per port and traffic direction.

This feature is supported only when HCA_CAP.ib_port_sniffer==1

29.38.1 ATTACH_TO_SNIFFER - Attach a QP to Port Sniffer

Table 2627 - ATTACH_TO_SNIFFER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h 04h
																op_mod																08h
traffic_sniffer																	qp_n															

Table 2627 - ATTACH_TO_SNIFFER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																0Ch

Table 2628 - ATTACH_TO_SNIFFER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	traffic_sniffer	Defines the traffic to be sniffed 0x0: Receive 0x1: Transmit	
	23:0	qpn	QP number	

Table 2629 - ATTACH_TO_SNIFFER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2630 - ATTACH_TO_SNIFFER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.38.2 DETACH_FROM_SNIFFER - Detach a QP from a Port Sniffer

Table 2631 - DETACH_FROM_SNIFFER Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
opcode																uid																00h	
																op_mod																04h	
traffic sniffer									qpn																								08h
																																	0Ch

Table 2632 - DETACH_FROM_SNIFFER Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	traffic_sniffer	Defines the traffic to be removed from sniffed QP 0x0: Receive 0x1: Transmit	
	23:0	qpn	QP number	

Table 2633 - DETACH_FROM_SNIFFER Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2634 - DETACH_FROM_SNIFFER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.39 GET_DROPPED_PACKET_LOG

This command returns logged packet header for dropped packet. Log should only be read after receiving the DROPPED_PACKET_LOGGED event (see Section 8.20.11.1, “DROPPED_PACKET_LOGGED Event,” on page 487. This command should only be called once per event received. No further events are sent until this command is called.

Dropped packet log consists of the first 128 bytes of a packet and the vendor specific drop syndrome.

Table 2635 - GET_DROPPED_PACKET_LOG Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2636 - GET_DROPPED_PACKET_LOG Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	64	reserved		

Table 2637 - GET_DROPPED_PACKET_LOG Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																00h																								
syndrome																																004h																								
																																08h-0Ch																								
packet_headers_log																																10h-8Ch																								
packet_syndrome																																90-CCh																								

Table 2638 - GET_DROPPED_PACKET_LOG Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		
10h-8Ch	1024	packet_headers_log	First 128 bytes of packet	
90h-CCh	512	packet_syndrome	For further information contact Mellanox technical support	

29.40 Diagnostic Counters Commands

Diagnostic commands are used to configure and enable diagnostic counters as explained in See Diagnostic Counters section

The following commands set up and maintain congestion.

Table 2639 - Diagnostic Counters Commands

Mnemonic	Command	Reference
QUERY_DIAGNOSTIC_PARAMS	Query Diagnostic parameters	“QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters” on page 2224
SET_DIAGNOSTIC_PARAMS	Set Diagnostic Parameters	“SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters” on page 2222

Table 2639 - Diagnostic Counters Commands (Continued)

Mnemonic	Command	Reference
QUERY_DIAGNOSTIC_COUNTERS	Query Diagnostic Counters	“QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters” on page 2226

29.40.1 SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters

SET_DIAGNOSTIC_PARAMS command configures the device to start/stop collecting diagnostics counters. In addition, it sets the counters list and the sampling period. The counters are fetched using the QUERY_DIAGNOSTIC_COUNTERS command.

Table 2640 - SET_DIAGNOSTIC_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223																																08h-..

Table 2641 - SET_DIAGNOSTIC_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-...	224	diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223	Diagnostic parameters context. Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223	

Table 2642 - SET_DIAGNOSTIC_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																																004h

Table 2642 - SET_DIAGNOSTIC_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch

Table 2643 - SET_DIAGNOSTIC_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2644 - DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
num_of_counters																										log_num_of_samples										00h
single	repetitive	sync	clear	on_demand	enable	tracer_dump											log_sample_period										04h									
																																08h-14h				
																counter_id[0]																18h				
																counter_id[1]																1Ch				
																...																20h-..				

Table 2645 - DIAGNOSTIC_PARAMS_CONTEXT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	num_of_counters	Number of valid counters to be captured in the list below.	
	7:0	log_num_of_samples	The number of samples to store on the device's sampling buffer is $2^{\text{log\_number\_of\_samples}}$. Range is $1 \dots \text{floor}(\text{DIAG\_CNT\_CAP.log\_max\_samples\_mul\_counters} / \text{DIAG\_CNT\_SET.num\_of\_counters})$ Note that modifying this field will cause resetting the sampling buffer.	
04h	31	single	Single sampling mode. Available only when HCA_CAP.single is '1'. Mutually exclusive with "repetitive" mode.	
	30	repetitive	Repetitive sampling mode. Available only when HCA_CAP.repetitive is '1' Mutually exclusive with "single" mode and on_demand mode.	
	29	sync	Use Synched Start counting on Diagnostics Counters that support sync operation. Sync is mutually exclusive with 'on demand'.	
	28	clear	Clear counters at the beginning of each period. Valid only for Synched Start Diagnostic Counters, only when sync is set to '1'. Clear is mutually exclusive with 'on demand'.	
	27	on_demand	If set, the device does not collect samples. Upon QUERY_DIAGNOSTIC command, the device will fetch current sample. 'on demand' mode is mutually exclusive with 'repetitive', 'clear' and 'sync' modes and cannot be set together	
	26	enable	0x1: Enable - enables diagnostic counting. 0x0: Disable - disables diagnostic counting.	
	25	tracer_dump	If set, the diagnostic counter report will be logged in the tracer buffer.	
	7:0	log_sample_period	The sample_period is $2^{\text{log\_sample\_period}}$ in device clocks. Range is diagnostic Capabilities Layout.Min_sample_period...35 (85 sec in 400 MHz clock)	
18h-...	15:0	counter_id[...]	Counter ids to capture.	

29.40.2 QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters

QUERY_DIAGNOSTIC_PARAMS command query the device configurations of collecting diagnostics counters.

Table 2646 - QUERY_DIAGNOSTIC_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2647 - QUERY_DIAGNOSTIC_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		

Table 2648 - QUERY_DIAGNOSTIC_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)																																08h-...

Table 2649 - QUERY_DIAGNOSTIC_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-...	224	diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)	Diagnostic parameters context. Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223	

29.40.3 QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters

QUERY_DIAGNOSTIC_COUNTERS command reports the counters samples with the sampling time stamp. The sample buffer is cyclic and when working in repetitive sampling mode might be overwritten if not queried often.

Table 2650 - QUERY_DIAGNOSTIC_COUNTERS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
num_of_samples																sample_index																08h
																																0Ch

Table 2651 - QUERY_DIAGNOSTIC_COUNTERS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:16	num_of_samples	The number of samples to return. Device might return up to the configured value HCA_CAP.log_number_of_samples. (See Section 29.3.4.10, “Debug Capabilities,” on page 1777) Note that the device can perform roll over when reaching number_of_samples, i.e. if the user asks for 256 samples starting from index 128, the result will be 128,...,255,0,...,127.	
	15:0	sample_index	The sample_index is the 1 st sample index. The sample index shall be in the range of 0... 2 ^{HCA_CAP.log_number_of_samples} . See Section 29.3.4.10, “Debug Capabilities,” on page 1777 .	

Table 2652 - QUERY_DIAGNOSTIC_COUNTERS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h

Table 2652 - QUERY_DIAGNOSTIC_COUNTERS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0Ch
diag_counter[0] (See Table 2654, “Diagnostic Counter Structure Layout,” on page 2227)																																10h-1Ch
diag_counter[1] (See Table 2654, “Diagnostic Counter Structure Layout,” on page 2227)																																20h-2Ch
...																																...

Table 2653 - QUERY_DIAGNOSTIC_COUNTERS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-...	128	diag_counter[...] (See Table 2654, “Diagnostic Counter Structure Layout,” on page 2227)	Diagnostic counter as described in Table 2654, “Diagnostic Counter Structure Layout,” on page 2227 The size of the counters list is num_of_samples*num_of_counters.	

Table 2654 - Diagnostic Counter Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
counter_id																sample_id																00h	
time_stamp[31:0]																																	04h
counter_value_h																																	08h
counter_value_l																																	0Ch

Table 2655 - Diagnostic Counter Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	counter_id	Counter identifier. 0 indicates invalid entry.	
	15:0	sample_id	Sample identifier	
04h	31:0	time_stamp[31:0]	LSB of time stamp.	

Table 2655 - Diagnostic Counter Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31:0	counter_value_h	MSB of counter value	
0Ch	31:0	counter_value_l	LSB of counter value	

29.41 Congestion Control Commands

The following commands set up and maintain congestion.

Table 2656 - Congestion control Commands

Mnemonic	Command	Reference
QUERY_CONG_STATUS	Query congestion status	“QUERY_CONG_STATUS - Query Congestion Status” on page 2228
MODIFY_CONG_STATUS	Modify congestion status	“MODIFY_CONG_STATUS - Query Congestion Status” on page 2230
QUERY_CONG_PARAMS	Query congestion parameters	“QUERY_CONG_PARAMS - Query Congestion parameters” on page 2232
MODIFY_CONG_PARAMS	Modify congestion parameters	“MODIFY_CONG_PARAMS - Modify Congestion Parameters” on page 2243
QUERY_CONG_STATISTICS	Query congestion Statistics	“QUERY_CONG_STATISTICS - Query Congestion Statistics” on page 2247

29.41.1 QUERY_CONG_STATUS - Query Congestion Status

Table 2657 - QUERY_CONG_STATUS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								priority				cong_protocol				08h
																																0Ch

Table 2658 - QUERY_CONG_STATUS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	

Table 2658 - QUERY_CONG_STATUS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	7:4	priority		
	3:0	cong_protocol	Congestion Protocol Status to Query 0x0: Spec_802_1Qau_Reaction_Point 0x1: R_RoCE_ECN_Reaction_Point 0x2: R_RoCE_ECN_Notification_Point	

Table 2659 - QUERY_CONG_STATUS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
status																																				00h
syndrome																																004h				
																																08h				
																																0Ch				
gen3_cc_negotiation_en																																				
tag_enable																																				
enable																																				

Table 2660 - QUERY_CONG_STATUS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

Table 2660 - QUERY_CONG_STATUS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	31	enable	For cong_protocol 802.1Qau Reaction Point (RP) - enable. For cong_protocol R-RoCE ECN Reaction Point (RP): cnp_receive_enable - when set RP is allowed to react to received CNP For cong_protocol R-RoCE ECN Notification Point (NP): ecn_receive_enable - when set NP is allowed to react to received ECN marked packets ('11').	
	30	tag_enable	Valid only for 802.1Qau Reaction Point (RP) congestion protocol.	
	29	gen3_cc_negotiation_en	If set, the scheduling of RoCE traffic works in the recommended congestion control mode with Mellanox 3rd generation NICs. Supported only when HCA_CAP.gen3_cc_negotiation==1.	

29.41.2MODIFY_CONG_STATUS - Query Congestion Status

Table 2661 - MODIFY_CONG_STATUS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset						
opcode																uid																00h						
																op_mod																04h						
																								priority		cong_proto- col		08h										
enable		tag_enable		gen3_cc_negotiation_en		gen3_cc_negotiation_mask																										0Ch						

Table 2662 - MODIFY_CONG_STATUS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	

Table 2662 - MODIFY_CONG_STATUS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
04h	15:0	op_mod		
08h	7:4	priority		
	3:0	cong_protocol	Congestion Protocol Status to Enable. Either QCN RP is enabled, or ECN RP + ECN NP is enabled. Attempting to enable ECN RP when QCN RP is already enabled, or the opposite is not allowed. 0x0: spec_802.1Qau Reaction Point 0x1: R_RoCE ECN Reaction Point 0x2: R_RoCE ECN Notification Point	
0Ch	31	enable	For cong_protocol 802.1Qau Reaction Point (RP) - enable. For cong_protocol R-RoCE ECN Reaction Point (RP): cnp_receive_enable - when set RP is allowed to react to received CNP For cong_protocol R-RoCE ECN Notification Point (NP): ecn_receive_enable - when set NP is allowed to react to received ECN marked packets ('11').	
	30	tag_enable	Valid only for 802.1Qau Reaction Point (RP) - enable.	
	29	gen3_cc_negotiation_en	If set, the scheduling of RoCE traffic works in the recommended congestion control mode with Mellanox 3rd generation NICs. Note that this a global configuration and not per prio/protocol. Valid only when gen3_cc_negotiation_mask==1	
	28	gen3_cc_negotiation_mask	If set, configuration of gen3_cc_negotiation_en is enabled, and all other configurations (enable and tag_enable) in this command will not be valid. Supported only when HCA_CAP.gen3_cc_negotiation==1.	

Table 2663 - MODIFY_CONG_STATUS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2664 - MODIFY_CONG_STATUS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.41.3 QUERY_CONG_PARAMS - Query Congestion parameters

Table 2665 - QUERY_CONG_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								cong_protocol				08h				
																																0Ch

Table 2666 - QUERY_CONG_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	3:0	cong_protocol	Congestion Protocol Parameters to Query 0x0: spec_802.1Qau Reaction Point (RP) 0x1: R-RoCE ECN Reaction Point (RP) 0x2: R-RoCE ECN Notification Point (NP)	

Table 2667 - QUERY_CONG_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2667 - QUERY_CONG_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
congestion_parameters_contexts																																10h-10Ch

Table 2668 - QUERY_CONG_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h-10Ch	2048	congestion_parameters_contexts	Parameters of Congestion control according to cong_protocol. if Cong_protocol ==0 - spec_802.1Qau Reaction Point (RP) (See Table 2669, “Congestion Control spec_802.1Qau RP Structure Layout,” on page 2233) if Cong_protocol ==1 - R-RoCE ECN Reaction Point (RP) (See Table 2671, “Congestion Control R-RoCE ECN RP Structure Layout,” on page 2235) if Cong_protocol ==2 - R-RoCE ECN Notification Point (NP) (See Table 2673, “Congestion Control R-RoCE ECN NP Structure Layout,” on page 2241)	

The tables below define the Congestion Control parameters for each congestion protocol.

Table 2669 - Congestion Control spec_802.1Qau RP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
rppp_max_rps																																10h
rpg_time_reset																																14h
rpg_byte_reset																																18h
rpg_threshold																																1Ch
rpg_max_rate																																20h
rpg_ai_rate																																24h

Table 2669 - Congestion Control spec_802.1Qau RP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
rpg_hai_rate																															28h	
rpg_gd																															2Ch	
rpg_min_dec_fac																															30h	
rpg_min_rate																															34h	
																															38h-FCCh	

Table 2670 - Congestion Control spec_802.1Qau RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	rppp_max_rps	Maximum number of RPs allowed for this CNPV on this port	
14h	31:0	rpg_time_reset	Time between rate increases if no CNMs received. Given in u-seconds.	
18h	31:0	rpg_byte_reset	Transmitted data between rate increases if no CNMs received. Given in Bytes	
1Ch	31:0	rpg_threshold	The number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states	
20h	31:0	rpg_max_rate	The maximum rate, in Mbits per second, at which an RP can transmit. Once this limit is reached, the RP rate limited is released and the flow is not rate limited any more.	
24h	31:0	rpg_ai_rate	The rate, in Mbits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE.	
28h	31:0	rpg_hai_rate	The rate, in Mbits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state.	
2Ch	31:0	rpg_gd	Upon CNM reception, flow rate is limited to $(1-Fb/Gd)^*$. CurrentRate. rpgGd is given as $\log_2(Gd)$, where Gd may only be powers of 2.	
30h	31:0	rpg_min_dec_fac	The minimum factor by which the current transmit rate can be changed by reception of a CNM. Value is given as a percentage (1-100).	
34h	31:0	rpg_min_rate	The minimum value, in Mbits per second, for rate to limit.	

Table 2671 - Congestion Control R-RoCE ECN RP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset 00h-08h
																																0Ch
																																10h
																																14h
																																18h
																																1Ch
																																20h
																																24h
																																28h
																																2Ch
																																30h
																																34h
																																38h
																																3Ch
																																40h
																																44h
																																48h
																																4Ch
																																50h
																																54h
																																58h

Table 2672 - Congestion Control R-RoCE ECN RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	27	clamp_tgt_rate	If set, whenever a CNP is processed, the target rate is updated to be the current rate.	
	23	clamp_tgt_rate_after_time_inc	When receiving an CNP, the target rate should be updated if the transmission rate was increased due to the timer, and not only due to the byte counter.	
	6	multi_qp_burst_decouple	When set multiple CC QPs subject to the same CC flow (i.e. same IP destination address) will attempt to avoid transmitting their data bursts at the same time, once CC allows them to transmit. Supported only when gen3_cc_negotiation_en is set and HCA_CAP.cc_roce_ecn_rp_burst_decouple	
	5	dynamic_g_en	When set, the dce_tcp_g will be updated dynamically according to the amount of received CNPs. The dce_tcp_g is decreased when the average amount of CNPs per CC QP per second is higher than dynamic_g_cnp_upper_threshold, and increased when the average is below dynamic_g_cnp_lower_threshold. The increments are done by dynamic_g_inc_step and the decrements are done by dynamic_g_dec_step. dce_tcp_g cannot cross beyond the range defined by max_dynamic_g and min_dynamic_g. While the feature is active, the static value is not used. When the feature is disabled, the dynamic value is discarded, and the static value is used.	
	4	dynamic_ai_en	When set the rpg_ai_rate will be updated dynamically according to the elapsed time since last CNP. When a CNP is detected, the field is reset to min_dynamic_ai. While no CNP is received, rpg_ai_rate will increase until it saturates at max_dynamic_ai. rpg_ai_rate is increased periodically, where each period its current value is multiplied by the number of periods passed since last CNP. While the feature is active, the static value is not used. When the feature is disabled, the dynamic value is discarded, and the static value is used.	
	3	dynamic_rtt_en	When set the dce_tcp_rtt will be updated dynamically according to the number of CC QPs. The dce_tcp_rtt are defined by dynamic_rtt_value <i> and dynamic_rtt_qp_threshold <i>. While the feature is active, the static value is not used. When the feature is disabled, the dynamic value is discarded, and the static value is used.	
	2:0	flow_classify_mode	Defines how a packet (data, CNP) is classified to a CC flow 0x0: BY_QPN - a flow is defined by the QP number 0x1: BY_DIP - a flow is defined by the IP destination address Supported only if HCA_CAP.cc_roce_ecn_rp_classify_mode is set.	

Table 2672 - Congestion Control R-RoCE ECN RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
14h	31:0	rpg_time_reset	Time between rate increases if no CNPs are received. Given in u-seconds.	
18h	31:0	rpg_byte_reset	Transmitted data between rate increases if no CNPs are received. Given in Bytes.	
1Ch	31:0	rpg_threshold	The number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states.	
20h	31:0	rpg_max_rate	The maximum rate, in Mbits per second, at which an RP can transmit. Once this limit is reached, the RP rate limited is released and the flow is not rate limited any more.	
24h	31:0	rpg_ai_rate	The rate, in Mbits per second, used to increase rpTarget-Rate in the RPR_ACTIVE_INCREASE.	
28h	31:0	rpg_hai_rate	The rate, in Mbits per second, used to increase rpTarget-Rate in the RPR_HYPER_INCREASE state.	
2Ch	31:0	rpg_gd	If an CNP is received, the flow rate is reduced at the beginning of the next rate_reduce_monitor_period interval to $(1 - \text{Alpha}/\text{Gd}) * \text{CurrentRate}$. rpgGd is given as $\log_2(\text{Gd})$, where Gd may only be powers of 2.	
30h	31:0	rpg_min_dec_fac	The minimum factor by which the current transmit rate can be changed when processing a CNP. Value is given as a percentage (1-100).	
34h	31:0	rpg_min_rate	The minimum value, in Mbits per second, for rate to limit.	
38h	31:0	max_dynamic_ai	Maximal value allowed for rpg_ai_rate to reach through dynamic updates. Value is given in Mbit per second. Valid only if dynamic_ai_en is set.	
3Ch	31:0	min_dynamic_ai	Reset value for rpg_ai_rate through dynamic updates. Upon detecting a CNP rpg_ai_rate is reset to this value. Must be smaller than max_dynamic_a. Value is given in Mbit per second. Valid only if dynamic_ai_en is set.	
40h	31:0	max_dynamic_g	Upper limit for dce_tcp_g during dynamic updates. The field will not increment beyond this value. Valid only if dynamic_g is set.	
44h	31:0	min_dynamic_g	Lower limit for dce_tcp_g during dynamic updates. The field will not decrement below this value. Valid only if dynamic_g is set.	
48h	31:0	dynamic_g_cnp_upper_threshold	When average number of CNPs per CC QP per second is above this value dce_tcp_g is decreased. Given in units of CNP per second. Valid only if dynamic_g is set.	

Table 2672 - Congestion Control R-RoCE ECN RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
4Ch	31:0	dynamic_g_cnp_lower_threshold	When average number of CNPs per CC QP per second is below this value dce_tcp_g is increased. Given in units of CNP per second. Valid only if dynamic_g is set.	
54h	31:0	rate_to_set_on_first_cnp	The rate that is set for the flow when a rate limiter is allocated to it upon first CNP received, in Mbps.	
58h	31:0	dce_tcp_g	Used to update the congestion estimator (alpha) once every dce_tcp_rtt microseconds, according to the equation: $\text{Alpha} = (\text{cnp\_received} * \text{dceTcpG}) + (1 - \text{dceTcpG}) * \text{alpha}$ dceTcpG is divided by 2^{10} . cnp_received is set to one if a CNP was received for this flow during period since the previous update and the current update.	
5Ch	31:0	dce_tcp_rtt	The time between updates of the alpha value in microseconds.	
60h	31:0	rate_reduce_monitor_period	The minimum time between 2 consecutive rate reductions for a single flow. Rate reduction will occur only if a CNP is received during the relevant time interval.	
64h	31:16	dynamic_g_inc_step	The amount by which dce_tcp_g should be increased every time the number of CNPs is measured below dynamic_g_cnp_lower_threshold. Value is given in the same units as dce_tcp_g. legal values are 0-1023. Valid only if dynamic_g is set.	
	15:0	dynamic_g_dec_step	The amount by which dce_tcp_g should be decreased every time the number of CNPs is measured above dynamic_g_cnp_upper_threshold. Value is given in the same units as dce_tcp_g. legal values are 0-1023. Valid only if dynamic_g is set.	
68h	31:0	initial_alpha_value	The initial value of alpha to use when receiving the first CNP for a flow. Expressed in a fixed point fraction of 2^{10} .	
6Ch	31:0	flow_definition_mode	Flow definition mode bit mask. Bit 0: per Destination Qp Bit 1: per Source Qp Bit 2: per FlowId Bit 3: per DMAC Bit 4: per Vlan Bit 5: per Prio	

Table 2672 - Congestion Control R-RoCE ECN RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
70h	31:0	dynamic_rtt_qp_thresh- old_0	dce_tcp_rtt will be set to dynamic_rtt_value_<i> if the number of CC QPs is greater or equal to dynamic_rtt_qp_ - threshold_<i>, for the highest index I that meets the condi- tion. dynamic_rtt_qp_threshold_<i> is expected to be smaller than dynamic_rtt_qp_threshold_<i+1>. Value of dynamic_rtt_value_<i> is given in microseconds. Valid only if dynamic_rtt_en is set.	
74h	31:0	dynamic_rtt_value_0		
78h	31:0	dynamic_rtt_qp_thresh- old_1		
7Ch	31:0	dynamic_rtt_value_1		
80h	31:0	dynamic_rtt_qp_thresh- old_2		
84h	31:0	dynamic_rtt_value_2		
88h	31:0	dynamic_rtt_qp_thresh- old_3		
8Ch	31:0	dynamic_rtt_value_3		
90h	31:0	dynamic_rtt_qp_thresh- old_4		
94h	31:0	dynamic_rtt_value_4		
98h	31:0	dynamic_rtt_qp_thresh- old_5		
9Ch	31:0	dynamic_rtt_value_5		

Table 2673 - Congestion Control R-RoCE ECN NP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-0Ch
flow_definition_mode																																10h
min_time_between_cnps																																14h

Table 2673 - Congestion Control R-RoCE ECN NP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																		cnp_dscp										cnp_prio mode	cnp_802 p_prio		18h	
																																1Ch-FCh

Table 2674 - Congestion Control R-RoCE ECN NP Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	flow_definition_mode	Flow definition mode bit mask. Bit 0: per Destination Qp Bit 1: per Source Qp Bit 2: per FlowId Bit 3: per DMAC Bit 4: per Vlan Bit 5: per Prio	
14h	31:0	min_time_between_cnps	Minimum time between sending cnps from the port in microseconds.	
18h	13:8	cnp_dscp	The DiffServ Code Point of the generated CNP for this port.	
	3	cnp_prio_mode	cnp prio mode. 0: use cnp_802p_prio 1: copy prio from received packet	
	2:0	cnp_802p_prio	The 802.1p priority value of the generated CNP for this port.	

29.41.4 MODIFY_CONG_PARAMS - Modify Congestion Parameters

Table 2675 - MODIFY_CONG_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																								cong_proto- col				08h				
field_select_802_1qau_rp_or_field_select_r_roce_rp_or_field_select_r_roce_np																												0Ch				
																												10h-1Ch				
congestion_parameters_context																												20h-11Ch				

Table 2676 - MODIFY_CONG_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	3:0	cong_protocol	Congestion Protocol Parameters to Modify 0x0 - spec_802.1Qau Reaction Point (RP) 0x1 - R-RoCE ECN Reaction Point (RP) 0x2 - R-RoCE ECN Notification Point (NP)	
0Ch	31:0	field_select_802_1qau_rp_or_field_select_r_roce_rp_or_field_select_r_roce_np	Specifies which parameters are selected to be modified. When Cong_protocol == 0: field_select 8021QauRP. See Table 2677, “FIELD_SELECT_SPEC_802.1Qau RP Structure Layout,” on page 2244. When Cong_protocol == 1: field_select R-RoCE RP. See Table 2679, “FIELD_SELECT_R-RoCE RP Structure Layout,” on page 2244. When Cong_protocol == 2: field_select R-RoCE NP. See Table 2682, “FIELD_SELECT_R-RoCE NP Structure Field Descriptions,” on page 2246.	

Table 2676 - MODIFY_CONG_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
20h-11Ch	2048	congestion_parameters_context	Parameters of Congestion control to be modified according to cong_protocol. if Cong_protocol ==0: spec_802.1Qau Reaction Point (RP) See Table 2669, “Congestion Control spec_802.1Qau RP Structure Layout,” on page 2233 if Cong_protocol ==1: R-RoCE ECN Reaction Point (RP) See Table 2671, “Congestion Control R-RoCE ECN RP Structure Layout,” on page 2235 if Cong_protocol ==2: R-RoCE ECN Notification Point (NP) See Table 2673, “Congestion Control R-RoCE ECN NP Structure Layout,” on page 2241	

Table 2677 - FIELD_SELECT_SPEC_802.1Qau RP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
field_select_8021QauRP																																00h

Table 2678 - FIELD_SELECT_SPEC_802.1Qau RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	field_select_8021QauRP	Bit Mask specifies which fields need to be modified Bit 2: rppp_max_rps Bit 3: rpg_time_reset Bit 4: rpg_byte_reset Bit 5: rpg_threshold Bit 6: rpg_max_rate Bit 7: rpg_ai_rate Bit 8: rpg_hai_rate Bit 9: rpg_gd Bit 10: rpg_min_dec_fac Bit 11: rpg_min_rate	

Table 2679 - FIELD_SELECT_R-RoCE RP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
field_select_R_RoCE_RP																																00h

Table 2680 - FIELD_SELECT_R-RoCE RP Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	field_select_R_Ro-CE_RP	Bit Mask specifies which fields need to be modified Bit 1: clamp_tgt_rate Bit 2: clamp_tgt_rate_after_time_inc Bit 3: rpg_time_reset Bit 4: rpg_byte_reset Bit 5: rpg_threshold Bit 6: rpg_max_rate Bit 7: rpg_ai_rate Bit 8: rpg_hai_rate Bit 9: rpg_min_dec_fac Bit 10: rpg_min_rate Bit 11: rate_to_set_on_first_cnp Bit 12: dce_tcp_g Bit 13: dce_tcp_rtt Bit 14: rate_reduce_monitor_period Bit 15: initial_alpha_value Bit 16: rpg_gd Bit 17: flow_classify_mode Bit 18: dynamic_rtt_en Bit 19: dynamic_rtt_qp_threshold_and_value_i- dynamic_rtt_qp_threshold_0/1/2/3/4/5 and dynamic_rtt_value_0/1/2/3/4/5 Bit 20: dynamic_ai_en Bit 21: dynamic_ai - max_dynamic_ai and min_dynamic_ai Bit 22: dynamic_g_en Bit 23: dynamic_g_params - dynamic_g_cnp_upper_threshold, dynamic_g_cnp_lower_threshold, dynamic_g_inc_ste, dynamic_g_dec_ste, max_dynamic_g and mix_dynamic_g Bit 24: multi_qp_burst_decouple	

Table 2681 - FIELD_SELECT_R-RoCE NP Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
field_select_R_RoCE_NP																																00h

Table 2682 - FIELD_SELECT_R-RoCE NP Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:0	field_select_R_Ro-CE_NP	Bit Mask specifies which fields need to be modified Bit 2: min_time_between_cnps Bit 3: cnp_dsep Bit 4: cnp_prio_mode/cnp_802p_prio others: reserved.	

Table 2683 - MODIFY_CONG_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2684 - MODIFY_CONG_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		

29.41.5 QUERY_CONG_STATISTICS - Query Congestion Statistics

Table 2685 - QUERY_CONG_STATISTICS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
clear																															08h	
																																0Ch

Table 2686 - QUERY_CONG_STATISTICS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31	clear	Clears counters after query is performed.	

Table 2687 - QUERY_CONG_STATISTICS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																								
status																																				00h																				
syndrome																																				004h																				
																																				08h-0Ch																				
cur_flows																																				10h																				
sum_flows																																				14h																				
cnp_ignored_high																																				18h																				
cnp_ignored_low																																				1Ch																				
cnp_handled_high																																				20h																				
cnp_handled_low																																				24h																				
cur_total_rate_high																																				28h																				
cur_total_rate_low																																				2Ch																				

Table 2688 - QUERY_CONG_STATISTICS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
10h	31:0	cur_flows	Number of flows limited	
14h	31:0	sum_flows	Sum of the number of flows limited over time (to calculate average number of flows limited) – a counter which is incremented once every X microseconds for every flow which is rate limited at the time.	
18h	31:0	cnp_ignored_high	Upper 32 Bits of cnp_ignored. Number of CNPs/CNMs ignored (i.e. due to FIFO being full or due to cache lines conflict)	
1Ch	31:0	cnp_ignored_low	Lower 32 Bits of cnp_ignored.	
20h	31:0	cnp_handled_high	Upper 32 Bits of cnp_handled. Number of CNPs/CNMs handled successfully	
24h	31:0	cnp_handled_low	Lower 32 Bits of cnp_handled.	
28h	31:0	cur_total_rate_high	Upper 32 bits of cur_total_rate. Current total rate of the rate limiters (sum of the speeds from all rate limiters) – in mbit/second	
2Ch	31:0	cur_total_rate_low	Lower 32 Bits of cur_total_rate.	
30h	31:0	sum_total_rate_high	Upper 32 Bits of sum_total_rate. Sum of total rate over time (like “sum of number of flows limited over time”, but summing over the rates of the flows). – in mbit/second	
34h	31:0	sum_total_rate_low	Lower 32 Bits of sum_total_rate.	
38h	31:0	min_total_rate_high	Upper 32 Bits of min_total_rate. Min total rate (need to be able to clean it from CR-space, keep the minimum value of the total rate since cleaning) – in mbit/second	
3Ch	31:0	min_total_rate_low	Lower 32 Bits of min_total_rate.	
40h	31:0	max_total_rate_high	Upper 32 Bits of max_total_rate. Max total rate (need to be able to clean it from CR-space, keep the maximum value of the total rate since cleaning) – in mbit/second	
44h	31:0	max_total_rate_low	Lower 32 Bits of max_total_rate.	
48h	31:0	time_stamp_high	Upper 32 Bits of time_stamp. Time stamp in microseconds	

Table 2688 - QUERY_CONG_STATISTICS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
4Ch	31:0	time_stamp_low	Lower 32 Bits of time_stamp.	
50h	31:0	accumulators_period	The value of X for the accumulating counters	
54h	31:0	ecn_marked_roce_packets_high	Upper 32 Bits of ecn_marked_roce_packets. ECN marked packets seen	
58h	31:0	ecn_marked_roce_packets_low	Lower 32 Bits of ecn_marked_roce_packets.	
5Ch	31:0	cnps_sent_high	Upper 32 Bits of cnps_sent. CNPs sent (can differ from ECN marked packets seen if the SX is not keeping up)	
60h	31:0	cnps_sent_low	Lower 32 Bits of cnps_sent.	
70h	31:0	roce_ecn_rp_dynamic_rtt_inc	Number of times CC QP threshold for dynamic RTT updates was crossed to a higher range. Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_rtt==1.	
74h	31:0	roce_ecn_rp_dynamic_rtt_dec	Number of times CC QP threshold for dynamic RTT updates was crossed to a lower range. Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_rtt==1.	
78h	31:0	roce_ecn_rp_dynamic_tcp_g_inc	Number of times dce_tcp_g was dynamically increased due to CNP reception below the average CNP threshold (The counter is incremented even if dce_tcp_g is at the maximal value). Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_g==1.	
7Ch	31:0	roce_ecn_rp_dynamic_tcp_g_dec	Number of times dce_tcp_g was dynamically decreased due to CNP reception above the average CNP threshold (The counter is incremented even if dce_tcp_g is at the minimal value). Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_g==1.	
80h	31:0	roce_ecn_rp_dynamic_ai_inc	Number of times rpg_ai_rate was increased due to absence of CNPs (The counter is incremented even if rpg_ai_rate reached the maximal value). Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_ai==1.	
84h	31:0	roce_ecn_rp_dynamic_ai_rst	Number of times rpg_ai_rate was reset to the minimal value due to CNPs (The counter is not incremented if rpg_ai_rate is already at the minimal value). Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_ai==1.	
88h	31:0	roce_ecn_rp_cur_dce_tcp_rtt	Current value of dce_tcp_rtt. Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_rtt==1.	
8Ch	31:0	roce_ecn_rp_cur_rpg_ai_rate	Current value of rpg_ai_rate. Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_ai==1.	

Table 2688 - QUERY_CONG_STATISTICS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
90h	31:0	roce_ecn_rp_cur_dce_tcp_g	Current value of dce_tcp_g. Valid only if HCA_CAP.cc_roce_ecn_rp_dynamic_g==1.	

29.42 VXLAN Commands

VXLAN UDP Destination ports are managed via ADD_VXLAN_UDP_DPORT and DELETE_VXLAN_UDP_DPORT. In addition l4_dport field must be updated in all relevant Flow Table's VXLAN rules via SET_FLOW_TABLE_ENTRY commands.

29.42.1 ADD_VXLAN_UDP_DPORT Command

Table 2689 - ADD_VXLAN_UDP_DPORT Command Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																vxlan_udp_port																0Ch

Table 2690 - ADD_VXLAN_UDP_DPORT Command Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod		
0Ch	15:0	vxlan_udp_port	VXLAN UDP destination port to add.	

Table 2691 - ADD_VXLAN_UDP_DPORT Command Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2692 - ADD_VXLAN_UDP_DPORT Command Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.42.2DELETE_VXLAN_UDP_DPORT Command

Table 2693 - DELETE_VXLAN_UDP_DPORT Command Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h
																vxlan_udp_port																0Ch

Table 2694 - DELETE_VXLAN_UDP_DPORT Command Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	15:0	vxlan_udp_port	VXLAN UDP destination port to add.	

Table 2695 - DELETE_VXLAN_UDP_DPORT Command Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2696 - DELETE_VXLAN_UDP_DPORT Command Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.43 Miscellaneous Commands

This chapter describes miscellaneous commands not related to any specific HCA resource.

29.43.1NOP Command

The NOP command creates a command completion event and has no other functions. See Section 8.20.14.1, “Command Interface Completion Event,” on page 501.

This command is useful for debug as well as for checking that command mechanism works.

Table 2697 - NOP Command Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2698 - NOP Command Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	31:0	reserved		

Table 2699 - NOP Command Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

29.44 EMC Specific Commands

Table 2700 - NOP Command Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

29.44.1 QUERY_RSV_RESOURCES

Table 2701 - QUERY_RSV_RESOURCES Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Ch

Table 2702 - QUERY_RSV_RESOURCES Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h-0Ch	31:0	reserved		

Table 2703 - QUERY_RSV_RESOURCES Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
				log2_rsvd_cqs								log2_rsvd_eqs								log2_rsvd_s-rqs								log2_rsvd_qps				08h
				log2_rsvd_psvs								log2_rsvd_m-keys								log2_rsvd_uars								log2_rsvd_mts				0Ch
																												log2_rsvd_bsfs				10h
																																14h-1Ch

Table 2704 - QUERY_RSV_RESOURCES Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	27:24	log2_rsvd_cqs	Log (base 2) of the number of CQs reserved for firmware use The reserved resources are numbered from 0 to $2^{\log2\_rsr\_vd\_cqs-1}$	
	19:16	log2_rsvd_eqs	Log (base 2) of the number of EQs reserved for firmware use The reserved resources are numbered from 0 to $2^{\log2\_rsr\_vd\_eqs-1}$	
	11:8	log2_rsvd_srqs	Log (base 2) of the number of SRQs reserved for firmware use The reserved resources are numbered from 0 to $2^{\log2\_rsr\_vd\_srqs-1}$	
	3:0	log2_rsvd_qps	Log (base 2) of the number of QPs reserved for firmware use The reserved resources are numbered from 0 to $2^{\log2\_rsr\_vd\_qps-1}$	

Table 2704 - QUERY_RSV_RESOURCES Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	27:24	log2_rsvd_psvs	Log (base 2) of the number of PSV indexes reserved for firmware use. The reserved resources are numbered from 0 to $2^{\log2\_rsrvd\_psvs}-1$.	
	19:16	log2_rsvd_mkeys	Log (base 2) of the number of MKeys reserved for firmware use. The reserved resources are numbered from 0 to $2^{\log2\_rsrvd\_mkeys}-1$.	
	11:8	log2_rsvd_uars	Log (base 2) of the number of UARs reserved for firmware use. The reserved resources are numbered from 0 to $2^{\log2\_rsrvd\_uars}-1$.	
	3:0	log2_rsvd_mmts	Log (base 2) of the number of MTTs reserved for firmware use. The reserved resources are numbered from 0 to $2^{\log2\_rsrvd\_mtts}-1$.	
10h	3:0	log2_rsvd_bsfs	Log (base 2) of the number of bsf indexes reserved for firmware use. The reserved resources are numbered from 0 to $2^{\log2\_rsrvd\_bsf}-1$.	
14h-1Ch	96	reserved		

29.44.2[EMC] CREATE_MTT - Create MTT Entries

CREATE_MTT is used to allocate and set multiple MTT entries. CREATE_MTT can be applied only to WQ, CQ and EQ MTTs. Command will allocate a number of MTTs as specified in the num_mmts field. MTT indexes will be consecutive, starting with the mtt_index specified in the field below.

Table 2705 - CREATE_MTT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset				
opcode																uid																00h				
																op_mod																04h				
num_mmts																										mtt_index_h										08h
mtt_index_l																														0Ch						
pas[0]																														10h-14h						
pas[1]																														18h-1Ch						
...																														...						

Table 2706 - CREATE_MTT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:16	num_mtt	Number of MTTs to create	
	6:0	mtt_index_h	MTT ICM address is $MTT\_base + 8 * mtt\_index$ (See Table 1804, “MTT Most Significant Bits Selection,” on page 1865 for constraints on MTT MSBs)	
0Ch	31:3	mtt_index_l		
10h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)	Physical address linked to first mtt	

Table 2707 - CREATE_MTT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
syndrome																														04h		
																														08h-0Ch		

Table 2708 - CREATE_MTT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	status		
04h	31:0	syndrome		

29.44.3[EMC] DESTROY_MTT - Destroy MTT Entries

DESTROY_MTT is used to de-allocate and set multiple MTT entries. DESTROY_MTT can be applied only to WQ, CQ and EQ MTTs. Command releases all valid MTTs in range (from mtt_index to mtt_index+num_mtt-1),

Table 2709 - DESTROY_MTT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
num_mtt																								mtt_index_h								08h
mtt_index_l																												0Ch				

Table 2710 - DESTROY_MTT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:16	num_mtt	Number of MTTs to destroy	
	6:0	mtt_index_h	MTT ICM address is $MTT\_base + 8 * mtt\_index$	
0Ch	31:3	mtt_index_l		

Table 2711 - DESTROY_MTT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-0Ch

Table 2712 - DESTROY_MTT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	status		
04h	31:0	syndrome		

29.44.4[EMC] QUERY_MTT - Query MTT Entries

QUERY_MTT is used to read multiple MTT entries. The command may fail if it tries to access non-existing MTTs or MTTs of different types (Data, KLMs or Control MTTs) in a single command.

Table 2713 - QUERY_MTT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
num_mtt																								mtt_index_h								08h
mtt_index_l																												0Ch				

Table 2714 - QUERY_MTT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
08h	31:16	num_mtt	Number of MTTs to be queried	
	6:0	mtt_index_h	MTT ICM address is MTT_base + 8 * mtt_index	
0Ch	31:3	mtt_index_l		

Table 2715 - QUERY_MTT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset																
status																																00h																
syndrome																																04h																
																																08h-0Ch																
pas[0]																																10h-14h																
pas[1]																																18h-1Ch																

Table 2715 - QUERY_MTT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
...																															..	

Table 2716 - QUERY_MTT Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	status		
04h	31:0	syndrome		
10h-...	64	pas[...] (See Table 1561, “Physical_Address_Structure (PAS) Layout,” on page 1665)	Physical address linked to first mtt	

29.44.5[EMC] ACCESS_CR

The ACCESS_CR command is used by the driver to access the device cr-space registers for device configuration etc.,

Table 2717 - ACCESS_CR Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
																																08h-0Cch
cr_address																																10h
data																																14h
mask																																18h

Table 2718 - ACCESS_CR Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod	Opcode modifier: 0 - write 1 - read	

Table 2718 - ACCESS_CR Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
10h	31:0	cr_address	Address must always be a multiple of 4 (command always accesses the entire dword)	
14h	31:0	data	For read - reserved For write - new register data	
18h	31:0	mask	For read - reserved. Setting bit in mask will cause new data to be written to corresponding bit. If mask is 0xffffffff new data will replace all old data (write) otherwise, read modify write will be performed. New data will only be written when corresponding bit is set in mask. For all other bits - old data will be preserved.	

Table 2719 - ACCESS_CR Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																												00h				
																														04h		
syndrome																																
data																														08h		
reserved																														0Ch		

29.44.6[EMC] MODIFY_SCHED_QUEUE

Table 2720 - ACCESS_CR Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h	31:0	data	For write - reserved For read - register data	
0Ch	31:0	reserved		

It should be called after INIT_HCA, but before WQEs are posted.

A single schedule queue object exists per port, SL and represents both the requestor and responder queues. In addition to the 16 SLs - VL15 has an additional queue. *Queue size, burst_size* and weight can be controlled separately for the requestor and responder sides. The queue offset is a shared property of both requestor and responder.

The Schedule Queue start offset can be controlled using the *sq_offset* field below. This allows better utilization of ICMC sets. *sq_offset* is in cache line granularity.

Schedule queue size is configured separately for requestor and responder. Size is in 4 byte granularity, and corresponds to the maximum number of QPs which can fit in a single SQ (each QP is represented by a 4 byte context). Please note, that if a schedule queue is meant to contain N QPs, $2^{\log_2 \text{size}}$ it must be greater or equal to $N + 16$. The additional 16 QP overhead is for hardware use.

Table 2721 - MODIFY_SCHED_QUEUE Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
opcode																uid																00h
																op_mod																04h
								port																vl 15	sl				08h			
reserved																																0Ch-1Ch
sched_queue_parameters (See Table 2723, “Schedule Queue Parameters Structure Layout,” on page 2263)																																20h-3Ch

Table 2722 - MODIFY_SCHED_QUEUE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, "UCTX - User Context," on page 539	
04h	15:0	op_mod	Opcode modifier: 0 - write 1 - read	
08h	23:16	port	port of schedule queue 0x0 - reserved 0x1-0x2 - port number 0x3-0xff - reserved	
	4	v15	v15 schedule queue	
	3:0	sl	SL of schedule queue Reserved when v15 bit is 1	
0Ch-1Ch	128	reserved		

Table 2722 - MODIFY_SCHED_QUEUE Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
20h-3Ch	256	sched_queue_parameters (See Table 2723, “Schedule Queue Parameters Structure Layout,” on page 2263)		

Table 2723 - Schedule Queue Parameters Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																				sq_offset								00h				
										log2_sq_size_re-sponder												log2_sq_size_re-questor					04h					
									burst_size_requestor																			08h				
									burst_size_responder																			0Ch				
						weight_responder												weight_requestor							10h							
								dwrr_weight																				14h				
																																18h-1Ch

Table 2724 - Schedule Queue Parameters Structure Field Descriptions

Offset	Bits	Name	Description	Access
0h	11:0	sq_offset	Schedule Queue buffer initial offset (in cache lines)	
04h	21:16	log2_sq_size_responder	Responder schedule queue size (in 4 byte granularity). legal values are: 5-24	
	5:0	log2_sq_size_requestor	Requestor schedule queue size (in 4 byte granularity). legal values are: 5-24	
08h	23:0	burst_size_requestor	Number of bytes allocated to requestor schedule queue per scheduling.(signed number, msb indicates sign). Note - actual requestor burst size = burst_size_requestor+device.burst_size	

Table 2724 - Schedule Queue Parameters Structure Field Descriptions

Offset	Bits	Name	Description	Access
0Ch	23:0	burst_size_responder	Number of bytes allocated to responder schedule queue per scheduling.(signed number, msb indicates sign). ote - actual responder burst size = burst_size_responder+device.burst_size	
10h	25:16	weight_responder	Relative weight of responder when schedule queue wins arbitration	
	9:0	weight_requestor	Relative weight of requestor when schedule queue wins arbitration	
14h	23:0	dwrr_weight	Number of tokens schedule queue receives upon reset event	
18h-1Ch	64	reserved		

Table 2725 - MODIFY_SCHED_QUEUE Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																04h
																																08h-1Ch
sched_queue_parameters																																20h-3Ch

Table 2726 - MODIFY_SCHED_QUEUE Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-1Ch	192	reserved		
20h-3Ch	256	sched_queue_parameters	See Table 2723, "Schedule Queue Parameters Structure Layout," on page 2263	

29.44.7 ARM_RESOURCE_DUMP_EVENT - ARM Resource Dump Event

For details, please refer to [Section 27.8, “Resource Dump,” on page 1062](#).

Table 2727 - ARM_RESOURCE_DUMP_EVENT Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset		
opcode																uid																00h		
																op_mod																04h		
																																08h		
vhca id valid																segment_type																0Ch		
																vhca_id																10h		
		index1																																14h
		index2																																18h
		num_of_obj1																num_of_obj2																1Ch

Table 2728 - ARM_RESOURCE_DUMP_EVENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:16	opcode		
	15:0	uid	Defines the user identifier for which the object was created. See Section 8.27.1, “UCTX - User Context,” on page 539	
04h	15:0	op_mod		
0Ch	29	vhca_id_valid	If set, then vhca_id field is valid. Otherwise dump resources on my vhca_id.	WO
	15:0	segment_type	See Section 27.8, “Resource Dump,” on page 1062 . If segment_type = menu segment, all segments supporting events will be armed; all other fields in this command will be ignored.	RW
10h	15:0	vhca_id	vhca_id where the resource is allocated.	WO
14h	31:0	index1	First object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	RW

Table 2728 - ARM_RESOURCE_DUMP_EVENT Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
18h	31:0	index2	Second object index to be dumped when supported by the object. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1.	RW
1Ch	31:16	num_of_obj1	The amount of objects to dump starting for index 1 SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj1_supports_all is set, the special value of 0xffff represents "all". When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents "active". The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj1 is "0".	RW
	15:0	num_of_obj2	The amount of objects to dump starting for index 2. SW shall read this field upon command done and shall provide it on the next call in case dump_more==1. Range is 0..0xffff0. When the segment's num_of_obj2_supports_all is set, the special value of 0xffff represents "all". When the segment's num_of_objx_supports_active is set, the special value of 0xfffe represents "active". The value of 0x0 and 0x1 are allowed even if the supported_num_of_obj2 is "0".	RW

Table 2729 - ARM_RESOURCE_DUMP_EVENT Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
syndrome																																004h
																																08h-0Ch

Table 2730 - ARM_MONITOR_COUNTER Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
00h	31:24	status		
04h	31:0	syndrome		
08h-0Ch	64	reserved		

30 ICMD Command Reference

30.1 Introduction

This chapter is a reference for all the configuration commands that can be executed via the ICMD command interface. For each command, the chapter describes the input and output Mailbox and certain restrictions (if applicable).

30.2 Conventions

- Reserved fields on input must be set to 0.
- Reserved fields on output must be ignored by software.
- Fields that appear void are considered reserved for every practical purpose.
- When a command is not completed successfully, output data is undefined, unless specified otherwise.

30.3 Command Summary

Table 2731 lists all the commands described in this chapter sorted by the opcode along with links to the respective sections describing the commands.

Note: [The range 0x2000 to 0x27FF is reserved for Switch FW.](#)

Note: [The range 0x2800 to 0x2FFF is reserved for NIC FW](#)

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 1 of 6)

Group	Opcode	Command Name	Link to Command Description
	0x1000	ICMD_OPCODE_MNG_MMINT	
	0x1002	ICMD_OPCODE_MNG_TEMP_HOOK	
Switch FW	0x2000-0x27FF	reserved	
NIC FW	0x2800-0x2FFF	reserved	
	0x6001	ICMD_OPCODE_PHY_UC_DO_CONTINUE	
	0x6002	ICMD_OPCODE_PHY_UC_DO_STEP	
	0x6003	ICMD_OPCODE_PHY_UC_SET_DEL_BP	
	0x6004	ICMD_OPCODE_PHY_UC_SET_GET_DATA	
	0x6005	ICMD_OPCODE_PHY_UC_GET_ARRAY_PROP	

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 2 of 6)

Group	Opcode	Command Name	Link to Command Description
	0x6006	ICMD_OPCODE_PHY_UC_GET_STATUS	
	0x6007	ICMD_OPCODE_PHY_ACTIVATE_RX - ADAP	
	0x6008	ICMD_OPCODE_PHY_UC_SET_GET - DATA_FIELD	
	0x6009	ICMD_OPCODE_PHY_UC_ACTIVATE_UC	
	0x600b	ICMD_OPCODE_PHY_GET_LINK_STATUS	
	0x600c	ICMD_OPCODE_PHY_OPEN_CLOSE_PORT	
	0x600d	ICMD_OPCODE_PHY_PLL_VARACTOR - CALIB	
	0x600e	ICMD_OPCODE_PHY_HALT_FSM	
	0x600f	ICMD_OPCODE_PHY_SET_GET_TX_SD	
	0x6010	ICMD_OPCODE_PHY_SET_GET_RX_SD	
	0x6011	ICMD_OPCODE_PHY_TRANS_OR_SYNC	
	0x6013	ICMD_OPCODE_PHY_GET - CLEAR_COUNTERS	
	0x6014	ICMD_OPCODE_PHY_UPDATE_CALI-B_VALS	
	0x6015	ICMD_OPCODE_PHY_GET_GRADE_INFO	
	0x6016	ICMD_OPCODE_PHY_SET_GET_PTAS	
	0x6017	ICMD_OPCODE_PHY_GET_PPAMP	
	0x6018	ICMD_OPCODE_PHY_GET_PPLL	
	0x6019	ICMD_OPCODE_PHY_GET_PPddb	
	0x8000	ICMD_OPCODE_ACCESS_ICM_LINE	
	0x8001	ICMD_OPCODE_ACCESS_ICM_CTX	
	0x8002	ICMD_OPCODE_ACCESS_HOST_MEM_PA	
	0x8003	ICMD_OPCODE_READ_Q_ENTRY	
	0x8004	ICMD_OPCODE_GET_IRISC_HEARTBEAT	
	0x8005	ICMD_OPCODE_GET_BOOT_STAGE	
	0x8006	ICMD_OPCODE_GET_LINK_LEDS	
MISC	0x8007	ICMD_OPCODE_QUERY_FW_INFO	Section 30.4.5, on page 2283

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 3 of 6)

Group	Opcode	Command Name	Link to Command Description
	0x8008	ICMD_OPCODE_READ_RX_SLICE_PKT	
	0x8009	ICMD_OPCODE_READ_G_RSE_DESC	
	0x8010	ICMD_OPCODE_READ_SX_WQ_BUFFER	
	0x8011	ICMD_OPCODE_READ_ICM_TLB_LINE	
	0x8012	ICMD_OPCODE_READ_DUMMY_ADDRESS	
	0x8013	ICMD_OPCODE_DEBUG_FW_TRACER	
	0x8014	ICMD_OPCODE_CABLE_PLUG_OUT_IN	
	0x8015	ICMD_OPCODE_GET_ICM_RES_PROP	
	0x8017	ICMD_OPCODE_GET_ICM_CTX_ADDRESS	
	0x8018	ICMD_OPCODE_CHECK_SQPC_STATUS	
	0x8019	ICMD_OPCODE_ALLOC_DEALLOC_RESOURCE	
	0x801a	ICMD_OPCODE_LOCK_UNLOCK	
	0x801b	ICMD_OPCODE_DB_RECOVERY	
	0x801c	ICMD_OPCODE_READ_ICMC_TAG	
	0x801e	ICMD_OPCODE_ACCESS_HOST_MEM_FULL	
	0x8020	ICMD_OPCODE_RW_GVMIX	
	0x8021	ICMD_OPCODE_ICMC_LOCKED_HISTOGRAM	
	0x8022	ICMD_OPCODE_DC_CONFIG	
	0x8023	ICMD_OPCODE_PUSH_DB	
	0x8024	ICMD_OPCODE_GET_OPEN_RESOURCES	
	0x8025	ICMD_OPCODE_ESW	
	0x8026	ICMD_OPCODE_ACCESS_REG_PMTU	
	0x8027	ICMD_OPCODE_ACCESS_REGISTER_PAOS	
	0x8028	ICMD_OPCODE_ACCESS_REG_PPCNT	
	0x8029	ICMD_OPCODE_ACCESS_REG_PTYS	
	0x802a	ICMD_OPCODE_ACCESS_REG_PVLC	

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 4 of 6)

Group	Opcode	Command Name	Link to Command Description
	0x8030	ICMD_OPCODE_ACCESS_REGISTER_SLRP	
	0x8031	ICMD_OPCODE_ACCESS_REGISTER_SLTP	
	0x8032	ICMD_OPCODE_ACCESS_REGISTER_SLRG	
	0x8033	ICMD_OPCODE_ACCESS_REGISTER_PTAS	
	0x8034	ICMD_OPCODE_ACCESS_REGISTER_P-PAMP	
	0x8100	ICMD_OPCODE_DEBUG_PRINT_ICM	
	0x8101	ICMD_OPCODE_GET_HEADER_CAPTURE	
	0x8102	ICMD_OPCODE_GEN_EQE	
	0x8200	ICMD_OPCODE_SET_GET_PORT_CNTR	
	0x8201	ICMD_OPCODE_GET_PF_CTX	
	0x8202	ICMD_OPCODE_GET_PI_CI	
	0x8300	ICMD_OPCODE_GET_FT_LIST	
	0x8301	ICMD_OPCODE_GET_FT_INFO	
	0x8302	ICMD_OPCODE_GET_FG_LIST	
	0x8303	ICMD_OPCODE_GET_FG	
	0x8304	ICMD_OPCODE_GET_FTE_LIST	
	0x8305	ICMD_OPCODE_GET_FTE	
	0x8306	ICMD_OPCODE_GET_STE_OPEN_RESOURCES	
	0x8307	ICMD_OPCODE_GET_STE_RESOURCES_LIST	
	0x8308	ICMD_OPCODE_READ_STE	
MISC	0x8400	ICMD_QUERY_CAPABILITY	Section 30.4.1, on page 2273
MISC	0x8402	ICMD_MH_SYNC	Section 30.4.3, on page 2280
MISC	0x8403	ICMD_MH_SYNC_STATUS	Section 30.4.4, on page 2283
INTERNAL	0x8404	ICMD_GET_FRC_CHALLENGE	Section 30.8.1, on page 2298
INTERNAL	0x8405	ICMD_SET_FRC	Section 30.8.2, on page 2301
	0x9000	ICMD_OPCODE_REG_ACCESS_DEBUG	
MISC	0x9001	ICMD_ACCESS_REGISTER	Section 30.4.2, on page 2276

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 5 of 6)

Group	Opcode	Command Name	Link to Command Description
	0x9002	ICMD_OPCODE_NVCONF_RW_TLV_DEBUG	
CONFIG	0x9003	ICMD_QUERY_VIRTUAL_MAC	Section 30.5.1, on page 2285
CONFIG	0x9004	ICMD_SET_VIRTUAL_MAC	Section 30.5.2, on page 2287
CONFIG	0x9005	ICMD_QUERY_WOL_ROL	Section 30.5.4, on page 2289
CONFIG	0x9006	ICMD_SET_WOL_ROL	Section 30.5.3, on page 2288
HP	0x9007	ICMD_OCBB_INIT	Section 30.7.2, on page 2295
HP	0x9008	ICMD_OCBB_QUERY_HEADER_STATUS	Section 30.7.3, on page 2296
HP	0x9009	ICMD_OCBB_QUERY_ETOC_STATUS	Section 30.7.4, on page 2297
HP	0x900A	ICMD_OCBB_SET_EVENT	Section 30.7.5, on page 2297
CONFIG	0x900B	ICMD_QUERY_VIRTUAL_PARAMETERS	Section 30.5.5, on page 2290
CONFIG	0x900C	ICMD_SET_VIRTUAL_PARAMETERS	Section 30.5.6, on page 2291
DEBUG	0x9020	ICMD_SET_DIAGNOSTIC_PARAMS	Section 30.6.1, on page 2292
DEBUG	0x9021	ICMD_QUERY_DIAGNOSTIC_PARAMS	Section 30.6.2, on page 2293
DEBUG	0x9022	ICMD_QUERY_DIAGNOSTIC_COUNTERS	Section 30.6.3, on page 2294
	0xa001	ICMD_OPCODE_FW_RESET	
	0xa004	ICMD_OPCODE_PMAOS_WRITE	
	0xa005	ICMD_OPCODE_PMAOS_QUERY	
	0xa099	ICMD_OPCODE_THERMAL_PROT_EN	
	0xa100	ICMD_OPCODE_MNG_SMBUS_MASTER_ACCESS	
	0xa102	ICMD_OPCODE_MNG_SMBUS_GW_FREQ	
	0xb000	ICMD_OPCODE_READ_COMMAND	
	0xc000	ICMD_OPCODE_READ_EQ4TYPE	
	0xc001	ICMD_OPCODE_READ_SPECIAL_QPS	
	0xc002	ICMD_OPCODE_SET_PORT_SNIFFER	
	0xc003	ICMD_OPCODE_GET_PORT_SNIFFER	
	0xc004	ICMD_OPCODE_WRITE_CC_LOG	
	0xc005	ICMD_OPCODE_CONFIG_CC	

Table 2731 - ICMD Commands List Sorted by Opcode (Sheet 6 of 6)

Group	Opcode	Command Name	Link to Command Description
	0xc006	ICMD_OPCODE_WRITE_DC_CNAK_LID	
	0xf000	ICMD_OPCODE_PHY_NEGO_CTRL	
	0xf001	ICMD_OPCODE_EYE_OPENER_READ	
	0xf002	ICMD_OPCODE_CLEAR_RXT_CHK_- CAUSES	
	0xf003	ICMD_OPCODE_SET_ITRACE	
HP	0xf004	ICMD_OPCODE_INIT_OCSD	Section 30.7.1, on page 2295
	0xf005	ICMD_OPCODE_PHY_VER_OVERWRITE	
	0xf006	ICMD_OPCODE_PCIE_TOOLS_POINTERS	
	0xf007	ICMD_OPCODE_CODE_COVERAGE	
	0xf008	ICMD_OPCODE_CODE_COVERAGE_READ	
	0xf009	ICMD_OPCODE_PCIE_VER_DATA	
	0xf00a	ICMD_OPCODE_STRESS_TEST	
	0xf00b	ICMD_OPCODE_STOP_TOGGLER	
	0xf010	ICMD_OPCODE_GREPPER	
	0xf011	ICMD_OPCODE_STANDBY	
	0xf012	ICMD_OPCODE_FLR_DEBUG	
	0xf013	ICMD_OPCODE_DCR_LISTS_TRVRS	
	0xf014	ICMD_OPCODE_DCR_LIST_NEXT	
	0xf015	ICMD_OPCODE_GENERAL_FAULT_INJEC- TOR	
	0xf016	ICMD_OPCODE_ACCESS_STEERING_- ROOT	
	0xf017	ICMD_OPCODE_GET_STEER- ING_TABLE_INFO	
	0xf018	ICMD_OPCODE_GET_GVMI_VIA_VSEC	
	0xf019	ICMD_OPCODE_XPORT_CONFIG	
	0xff60	ICMD_OPCODE_CABLE_INFO	
	0xff61	ICMD_OPCODE_GEN_ASSERT	

30.4 Miscellaneous Commands (MISC)

30.4.1 ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities

The ICMD_QUERY_CAPABILITY command is used to expose the ICMD interface capabilities to the driving SW tool. These capabilities allow the tool to a-priory enable/disable flows instead of attempting to activate them and receive an *Invalid Command* response at run time.

Table 2732 - ICMD_QUERY_CAPABILITY Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																capability_group												00h				

Table 2733 - ICMD_QUERY_CAPABILITY Input Mailbox Field Description

Offset	Bits	Name	Description
00h	15:0	capability_group	Group of capabilities. 0x0: General_capability 0x1: Debug_Capability other values are reserved.

Table 2734 - ICMD_QUERY_CAPABILITY Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
icmd_capability																																00h-04h

Table 2735 - ICMD_QUERY_CAPABILITY Output Mailbox Field Description

Offset	Bits	Name	Description
00h-04h	64	icmd_capability	ICMD capabilities according to capability_group: 0x0: General_capability - see Table 2736, “ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout,” on page 2274 0x1: Debug_capability - see Table 2738, “ICMD_QUERY_CAPABILITY Debug Capability Output Mailbox Layout,” on page 2276 Other values are reserved.

Table 2736 - ICMD_QUERY_CAPABILITY General Capability Output Mailbox Layout

Offset	00h	04h
0		wol_p
1		wol_u
2		wol_m
3		wol_b
4		wol_a
5		wol_g
6		wol_s
7		
8		rol_g
9		rol_s
10		fpga
11		
12		
13		
14		
15		
16	ncfg_reg	
17	cwcam_reg	
18	sbcam_reg	
19		
20	capability_groups	
21	strs_cap	
22	virtual link down	
23	icmd_exmb	
24	capi	
25	qcam_reg	
26	mcam_reg	
27	pcam_reg	
28	mlh_sync	
29	allow_icmd_access_reg_on_all_registers	
30	fw_info_psid	
31	nv_access	

num_of_diagnostic_counters

Table 2737 - ICMD_QUERY_CAPABILITY General Capability Output Mailbox Field Description

Offset	Bits	Name	Description
00h	31	nv_access	When set, ICMD_ACCESS_REGISTER supports the following registers: - NVDA - NVDI - NVIA - NVQC
	30	fw_info_psid	When set, ICMD_QUERY_FW_INFO returns the <i>psid</i> field.
	29	allow_icmd_access_reg_on_all_registers	If set, ICMD_ACCESS_REGISTER supports every register. (in the past it supported some of them).
	28	mh_sync	When set, multi-host synchronization through the device is supported. See Section 30.4.3, “ICMD_MH_SYNC - Multi-Host Synchronization,” on page 2280.
	27	pcam_reg	If set, PCAM register is supported. See Table 952, “PCAM - Ports Capabilities Mask Register Layout,” on page 1146
	26	mcam_reg	If set, MCAM register is supported. See Table 1546, “MCAM - Management Capabilities Mask Register Layout,” on page 1643.
	25	qcam_reg	If set, QCAM register is supported. See Table 785, “QCAM - QoS Capabilities Mask Register Layout,” on page 977.
	24	capi	If set, CAPI is supported.
	23	icmd_exmb	When set, the device supports using a memory buffer as a mailbox for the ICMD, instead of the normal ICMD Mailbox.
	22	virtual_link_down	When set, the device supports emulating link down for all the associated functions using “ICMD_SET_VIRTUAL_PARAMETERS - Set Device Virtual Parameters”
	21	strs_cap	When set, STRS_CAP register is supported
	20	capability_groups	When set, the command supports capability groups in addition to the General Capability group
	18	sbcam_reg	If set, SBCAM register is supported. See Table 1324, “SBCAM - Shared Buffer Capabilities Mask Register Layout,” on page 1490
	17	cwcaml_reg	If set, CWCAM register is supported. Table 1266, “CWCAM - Congestion WRED Capabilities Mask Register Layout,” on page 1447
	16	ncfg_reg	If set, NCFG register is supported.

Table 2737 - ICMD_QUERY_CAPABILITY General Capability Output Mailbox Field Description

Offset	Bits	Name	Description
04h	31:16	num_of_diagnostic_counters	Indicates number of diagnostic counters supported by device. See Debug Capabilities for more information
04h	10	fpga	If set, FPGA device is supported by the device. See TBD
	9	rol_s	If set, reset on secured magic packet is supported.
	8	rol_g	If set, reset on magic packet is supported.
	6	wol_s	If set, wake on secured magic packet is supported.
	5	wol_g	If set, wake on magic packet is supported.
	4	wol_a	If set, wake on ARP is supported.
	3	wol_b	If set, wake on broadcast packet is supported.
	2	wol_m	If set, wake on multicast packet is supported.
	1	wol_u	If set, wake on unicast packet is supported.
	0	wol_p	If set, wake on phy activity is supported.

Table 2738 - ICMD_QUERY_CAPABILITY Debug Capability Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
debug_capability																																00h

Table 2739 - ICMD_QUERY_CAPABILITY Debug Capability Output Mailbox Field Description

Offset	Bits	Name	Description
00h	31:0	debug_capability	See Table 1599, “Debug Capabilities Layout,” on page 1777 The actual supported structure size is defined by the ICMD mailbox size and General Capability <i>num_of_diagnostic_counters</i>

30.4.2 ICMD_ACCESS_REGISTER - Register Access

The ICMD_ACCESS_REGISTER command is a method for issuing commands (named ‘Registers’ in this context) to the device through the ICMD interface. It is similar to the command interface in its function.

Registers are constructed of fields of the following types:

- Index – used as an index into an internal table. For example, port may be an index in a register accessing the related properties of a specific port as indicated by the user in the **port** field. A field indicated as Index must be set for both Query and Write operations' input mailbox.
- RO – read-only field used to query the capability or state of the device. This field cannot be set by the software, so it is reserved in output mailbox of Write operation. Such a field is also reserved in input mailbox of both Query and Write operations.
- RW – read/write field used to set and query. Such a field should be set in Write operations input mailbox and is reserved in Query operation input mailbox.
- WO – write-only field used to activate an internal state machine (command). Such a field should be set in Write operation input mailbox and is reserved in Query operation input mailbox. It is also reserved for both Query and Write operations' output mailbox.

Note that, for ICMD_ACCESS_REGISTER, the status field reported in [Table 359, “ICMD Control Layout,” on page 620](#) indicates the success of the command. The success of the register access is indicated by the status field in [Table 2744, “ICMD_ACCESS_REGISTER Output Mailbox Layout,” on page 2279](#). For example, if the register holds an invalid parameter value, the register access will fail, while the command itself will succeed.

The field types are summarized in [Table 2740, “ICMD_ACCESS_REGISTER Field Types Summary”](#).

Table 2740 - ICMD_ACCESS_REGISTER Field Types Summary

Field Type	Query		Write	
	Input Mailbox	Output Mailbox	Input Mailbox	Output Mailbox
Index	Must be set	Valid (same vale as in the request)	Must be set	Valid (same vale as in the request)
RO	Reserved	Valid	Reserved	Reserved
RW	Reserved	Valid	Must be set	Valid
WO	Reserved	Reserved	Must be set	Reserved

The defined registers, their IDs and their supported access methods are listed in [Table 2741](#), “[ICMD_ACCESS_REGISTER Registers Summary](#)”.

Table 2741 - ICMD_ACCESS_REGISTER Registers Summary

Register Name	ID	Query	Write	Reference
MNVDA – Non volatile data access	0x9024	v	v	See Table 1326, “Management and Peripherals Control Register Summary,” on page 1492
MNVDI –NV-data Invalidate	0x9025		v	
NVCT – Non volatile commit	0x9032		v	
MNVIA – Non volatile invalidate all	0x9033		v	
MNVQC – Non-volatile query capability	0x9030	v		

Table 2742 - ICMD_ACCESS_REGISTER Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
constant_1				constant_2																								00h				
register_id																	method						constant_3						04h			
																																08h-0Ch
constant_4				len																								10h				
register_data[0]																																14h
register_data[1]																																18h
...																																::

Table 2743 - ICMD_ACCESS_REGISTER Input Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	31:27	constant_1	Must be set 0x1.
	26:16	constant_2	Must be set 0x4.

Table 2743 - ICMD_ACCESS_REGISTER Input Mailbox Field Descriptions

Offset	Bits	Name	Description
04h	31:16	register_id	ID of the register to access. See Table 2741, “ICMD_ACCESS_REGISTER Registers Summary,” on page 2278.
	14:8	method	0x01: Query 0x02: Write
	7:0	constant_3	Must be set 0x1.
10h	31:27	constant_4	Must be set to 0x3.
	26:16	len	Register length given in DWORDs. Includes the length of the <i>register_data</i> field + 1.
14h-...	31:0	register_data[...]	The register referenced by the <i>register_id</i> .

Table 2744 - ICMD_ACCESS_REGISTER Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset						
																status																00h						
register_id																																						04h
																																08h-0Ch						
					len																											10h						
register_data[0]																																14h						
register_data[1]																																18h						
...																																...						

Table 2745 - ICMD_ACCESS_REGISTER Output Mailbox Field Description

Offset	Bits	Name	Description
00h	14:8	status	Register access delivery status. 0x0: Good - Operation completed 0x1: Device_busy - Cannot perform the operation at the moment 0x2: Version_not_supported - [Internal] 0x3: Bad_command_TLV_layout - [Internal] Unknown TLV. 0x4: Register_not_supported 0x5: Bad_command_Class_layout - [Internal] Class not supported. 0x6: Method_not_supported 0x7: Bad_parameter - e.g. port out of range 0x8: Resource_not_available 0x9: Message_Receipt_Acknowledgment - will return answer later. [Internal] 0xA: retry_emad_task_in_next_FW_loop - [Internal] 0x20: Unsupported_configuration - [internal] FW may refuse a SW2FW TLV 0x21: Flash_erase_count_exceeded 0x22: Fatal_error - [Internal] config partition corrupted 0x24: Register_length_too_small 0x70: Internal_Error 0x7F: Drop - [Internal] Other values are reserved for future error codes and should be handled as if the command failed for an unknown reason.
04h	31:16	register_id	ID of the register to access. See Table 2741, “ICMD_ACCESS_REGISTER Registers Summary,” on page 2278.
10h	26:16	len	Register length given in DWORDs. Includes the length of the <i>register_data</i> field + 1.
14h-...	31:0	register_data[...]	The register referenced by the <i>register_id</i> . Valid only for <i>Query method</i> .

30.4.3 ICMD_MH_SYNC - Multi-Host Synchronization

The device implements a synchronization mechanism among tools running on different hosts on a multi-host server. The mechanism is defined by the a finite state machine (FSM) with the following states:

1. IDLE - This is the initial state of the FSM. No synchronization operation is in progress. This state is exited once a host issues a state change request to GET_READY state.
2. GET_READY - Synchronization operation is in progress, waiting for all hosts to acknowledge the process by re-asserting the state to GET_READY. Once all hosts acknowledge the state, the FSM transitions to GO state.
3. GO - Synchronization operation is done, waiting for all hosts to acknowledge the completion of the operation by re-asserting the state to GO.

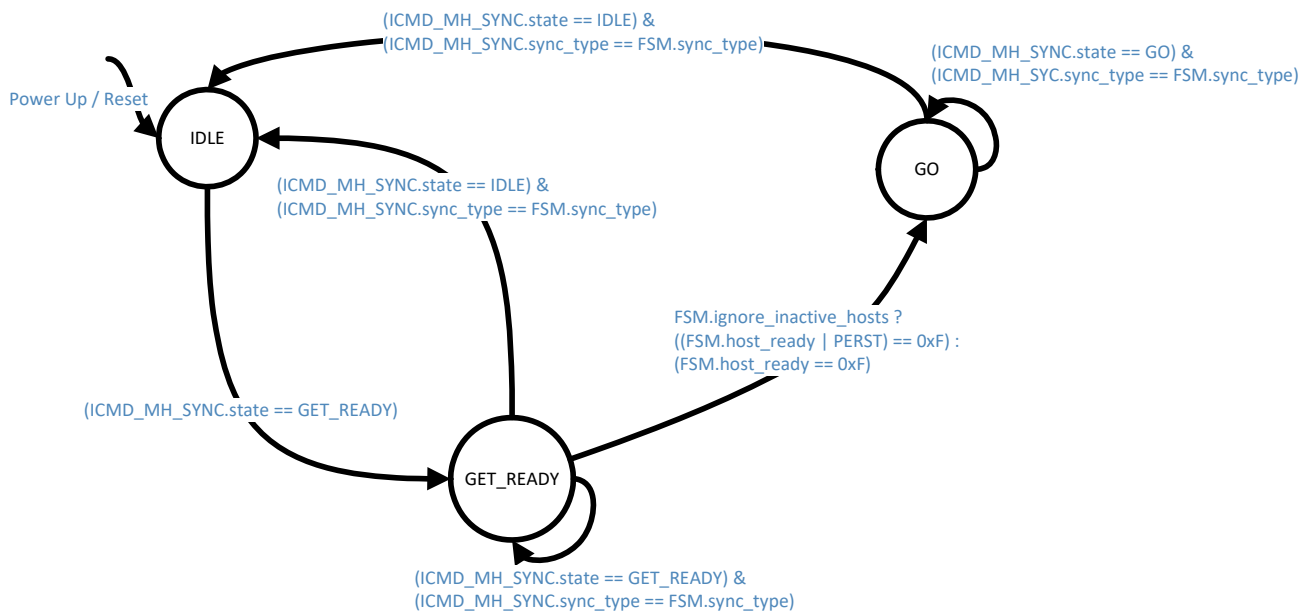
When a tool starts a synchronized operation, it samples the FSM, and if it is in IDLE state, the tool will modify the state to GET_READY.

When a tool notices that the FSM state is GET_READY, it should perform the required preliminary operations for the specific synchronized operation, and then re-assert the FSM state to GET_READY. When FSM state changes to GO, the tool should complete the required operations for the specific synchronized operation. The tool may be required to frequently poll the FSM when it is waiting for it to move to GO state.

Once the operation is done, the tool should re-assert the state to GO, to inform the FSM of the completion of the operation. Once all hosts acknowledge the completion of the operation, any of them may transition the FSM back to IDLE state.

A Synchronization operation may be canceled at any time by setting the FSM state back to IDLE.

Figure 117: Host Synchronization FSM



The support for multi-host synchronization is indicated by *mh_sync* field in “[ICMD_QUERY - CAPABILITY - Get ICMD Interface Capabilities](#)” command.

The FSM is controlled using the ICMD_MH_SYNC command. The next state is asserted in *state* field, and the type of synchronization operation is specified in *sync_type* field. If some of the hosts are not active, they can be excluded from the synchronization operation by setting the *ignore_inactive_host* field.

The current state may be queried using the “[ICMD_MH_SYNC_STATUS - Multi-Host Synchronization Status](#)” command. In addition to the current state, this command indicates the acknowledgment status of each host in *host_ready* field. For GET_READY state, a set bit indicates that the appropriate host has re-asserted the GET_READY state. For GO state, a set bit indicates that the appropriate host has not re-asserted the GO state, yet. For IDLE state a set bit is an indication for a failed synchronized operation.

Note: After a tool performs the FW_RESET synchronized operation, it should continue polling the FSM state. If the state remains GO, then the FW reset failed.

Table 2746 - ICMD_MH_SYNC Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ignore_inactive_host																sync_type												state				00h

Table 2747 - ICMD_MH_SYNC Input Mailbox Field Description

Offset	Bits	Name	Description
00h	31	ignore_inactive_host	When set, hosts in reset state (PERST# low) are ignored (not synchronized)
	15:8	sync_type	The operation currently being synchronized: 0x1: FW_RESET other values are reserved
	3:0	state	Next state for the synchronization state machine: 0x0: IDLE 0x1: GET_READY 0x2: GO Other values are reserved

Table 2748 - ICMD_MH_SYNC Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
ignore_inactive_host																sync_type												state				00h
																															host_ready	04h
start_uptime																																08h

Table 2749 - ICMD_MH_SYNC Output Mailbox Field Description

Offset	Bits	Name	Description
00h	31	ignore_inactive_host	When set, hosts in reset state (PERST# low) are ignored (not synchronized)
	15:8	sync_type	The operation currently being synchronized: 0x1: FW_RESET other values are reserved
	3:0	state	Current state of the synchronization state machine: 0x0: IDLE 0x1: GET_READY 0x2: GO Other values are reserved
04h	3:0	host_ready	Bit per host. When set, indicates the appropriate host has already issued a state change command to the current state.
08h	31:0	start_uptime	NIC uptime (time since last reset) in microseconds, at the moment the synchronization state machine entered the GET_READY state.

30.4.4 ICMD_MH_SYNC_STATUS - Multi-Host Synchronization Status

See description in [“ICMD_MH_SYNC - Multi-Host Synchronization”](#) on page 2280.

This Command has no Input mailbox. The output mailbox is given in [Table 2748, “ICMD_MH_SYNC Output Mailbox Layout”](#).

30.4.5 ICMD_QUERY_FW_INFO - Get device firmware information

This command retrieves the current FW version burnt to the device along with the adapter identifier PSID.

The support for reporting the PSID is indicated by *fw_info_psid* field in [“ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities”](#).

Table 2750 - ICMD_QUERY_FW_INFO Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
fw_version (See Table 2752, “FW Version Layout,” on page 2284)																																00h-0Ch
																hash_signature																10h
psid																																14h-20h

Table 2750 - ICMD_QUERY_FW_INFO Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																24h-0FCCh

Table 2751 - ICMD_QUERY_FW_INFO Output Mailbox Field Descriptions

Offset	Bits	Name	Description
00h-0Ch	128	fw_version (See Table 2752, “FW Version Layout,” on page 2284)	FW version.
10h	15:0	hash_signature	Signature verifying FW integrity.
14h-20h	128	psid	See <i>psid</i> field in “QUERY_HCA_CAP – Query Device Capabilities” on page 1667

Table 2752 - FW Version Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
major																																00h
minor																subminor																04h
second								minute								hour																08h
year																month								day								0Ch

Table 2753 - FW Version Field Descriptions

Offset	Bits	Name	Description
00h	31:16	major	FW major version.
04h	31:16	minor	FW minor version.
	15:0	subminor	FW subminor version.
08h	31:24	second	FW timestamp - seconds.
	23:16	minute	FW timestamp - minutes.
	15:8	hour	FW timestamp - hours.

Table 2753 - FW Version Field Descriptions

Offset	Bits	Name	Description
0Ch	31:16	year	FW timestamp - year. Each 4 bits is a decimal digit (in hex). For Example, year 1980 will be exposed as: bits[31:28] = 1 bits[27:24] = 9 bits[23:20] = 8 bits[19:16] = 0
	15:8	month	FW timestamp - month.
	7:0	day	FW timestamp - day.

30.5 Configuration Commands (CONFIG)

30.5.1 ICMD_QUERY_VIRTUAL_MAC - Get Device MAC Address

This command is used to retrieve the factory defined MAC address of the device, and its currently configured MAC address.

Table 2754 - ICMD_QUERY_VIRTUAL_MAC Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
virtual_mac_en	mac_aux_v	~~virtual_node_guid_en~~		pcie_index				pf_index																							00h		
		permanent_mac (See Table 33, “MAC Address Layout,” on page 293)																															04h-08h
		virtual_mac (See Table 33, “MAC Address Layout,” on page 293)																															0Ch-10h
																																	14h-1Ch
		factory_port_guid																															20h-24h
factory_node_guid																															28h-2Ch		

Table 2754 - ICMD_QUERY_VIRTUAL_MAC Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																30h-0FCh

Table 2755 - ICMD_QUERY_VIRTUAL_MAC Output Mailbox Field Description

Offset	Bits	Name	Description
00h	31	virtual_mac_en	When set, a virtual MAC was set to replace the Factory defined MAC address
	30	mac_aux_v	During standby mode (if supported), the device restores the Factory defined MAC. When mac_aux_v is set, that will happen even if auxiliary power is supplied.
	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register
04h-08h	64	permanent_mac (See Table 33, “MAC Address Layout,” on page 293)	Factory defined MAC address for this PF.
0Ch-10h	64	virtual_mac (See Table 33, “MAC Address Layout,” on page 293)	Virtual MAC address currently configured to override the permanent_mac . Valid if virtual_mac_en is set.
20h-24h	64	factory_port_guid	Factory Port GUID address for this PF. 0 means NA
28h-2Ch	64	factory_node_guid	Factory Port NODE address for this PF. 0 means NA

30.5.2 ICMD_SET_VIRTUAL_MAC - Set Device MAC Address

This command is used to override the factory defined MAC address of the device by setting a virtual MAC address.

Table 2756 - ICMD_SET_VIRTUAL_MAC Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_mac_en	mac_aux_v			pcie_index				pf_index																								00h
																															04h-08h	
				virtual_mac (See Table 33, “MAC Address Layout,” on page 293)																											0Ch-10h	
																															14h-0FCh	

Table 2757 - ICMD_SET_VIRTUAL_MAC Input Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	31	virtual_mac_en	When set, a virtual MAC is set to replace the Factory defined MAC address
	30	mac_aux_v	During standby mode (if supported), the device restores the Factory defined MAC. When mac_aux_v is set, that will happen even if auxiliary power is supplied.
	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register
0Ch-10h	64	virtual_mac (See Table 33, “MAC Address Layout,” on page 293)	Virtual MAC address currently configured to override the permanent_mac . Valid if virtual_mac_en is set. value 0x0 will restore the factory-defined MAC.

30.5.3 ICMD_SET_WOL_ROL - Set Wake/Reset-On-LAN

This command is used to configure the Wake-on-LAN and Reset-on-LAN modes. See [“Wake On LAN \(WOL\)” on page 246](#) and [“Reset On LAN \(ROL\)” on page 246](#),

Table 2758 - ICMD_SET_WOL_ROL Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				pcie_index				pf_index																								00h
																																00h-04h
rol_mode_valid	wol_mode_valid															rol_mode							wol_mode							08h		
																																0Ch-0FCh

Table 2759 - ICMD_SET_WOL_ROL Input Mailbox Field Description

Offset	Bits	Name	Description
00h	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register

Table 2759 - ICMD_SET_WOL_ROL Input Mailbox Field Description

Offset	Bits	Name	Description
08h	31	rol_mode_valid	Set to 0x1 to modify <i>rol_mode</i> .
	30	wol_mode_valid	Set to 0x1 to modify <i>wol_mode</i> .
	15:8	rol_mode	Bitmask field indicates which Reset on LAN modes to be enabled/disabled.(valid when rol_mode_valid ==1). The support for each mode is indicated by a field in “ ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities ”. Bit 0: reset_on_magic_packet - Supported only when rol_g==1. Bit 1: reset_on_secured_magic_packet - Supported only when rol_s==1.
	7:0	wol_mode	Bitmask field indicates which Wake on LAN modes to be enabled/disabled (valid when wol_mode_valid == 1). The support for each mode is indicated by a field in “ ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities ”. Bit 1: wake_on_secured_magic_packet - Supported only when ICMD_CAP.wol_s==1. Bit 2: wake_on_magic_packet - Supported only when ICMD_CAP.wol_g==1. Bit 3: wake_on_ARP - Supported only when ICMD_CAP.wol_a==1. Bit 4: wake_on_broadcast_packet - Supported only when ICMD_CAP.wol_b==1. Bit 5: wake_on_multicast_packet - Supported only when ICMD_CAP.wol_m==1. Bit 6: wake_on_unicast_packet - Supported only when ICMD_CAP.wol_u==1. Bit 7: wake_on_phy_activity - Supported only when ICMD_CAP.wol_p==1.

30.5.4 ICMD_QUERY_WOL_ROL - Get Wake/Reset-On-LAN

This command is used to read the current Wake-on-LAN and Reset-on-LAN modes. See “[Wake On LAN \(WOL\)](#)” on page 246 and “[Reset On LAN \(ROL\)](#)” on page 246

Table 2760 - ICMD_QUERY_WOL_ROL Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
				pcie_index				pf_index																				00h				
																																00h-04h
																rol_mode								wol_mode								08h

Table 2761 - ICMD_QUERY_WOL_ROL Output Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register
08h	15:8	rol_mode	Bitmask field indicates which Reset on LAN modes to be enabled/disabled (valid when rol_mode_valid ==1). The support for each mode is indicated by a field in “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities” . Bit 0: reset_on_magic_packet - Supported only when rol_g==1. Bit 1: reset_on_secured_magic_packet - Supported only when rol_s==1.
	7:0	wol_mode	Bitmask field indicates which Wake on LAN modes to be enabled/disabled (valid when wol_mode_valid == 1). The support for each mode is indicated by a field in “ICMD_QUERY_CAPABILITY - Get ICMD Interface Capabilities” . Bit 1: wake_on_secured_magic_packet - Supported only when ICMD_CAP.wol_s==1. Bit 2: wake_on_magic_packet - Supported only when ICMD_CAP.wol_g==1. Bit 3: wake_on_ARP - Supported only when ICMD_CAP.wol_a==1. Bit 4: wake_on_broadcast_packet - Supported only when ICMD_CAP.wol_b==1. Bit 5: wake_on_multicast_packet - Supported only when ICMD_CAP.wol_m==1. Bit 6: wake_on_unicast_packet - Supported only when ICMD_CAP.wol_u==1. Bit 7: wake_on_phy_activity - Supported only when ICMD_CAP.wol_p==1.

30.5.5 ICMD_QUERY_VIRTUAL_PARAMETERS - Get Device Virtual Parameters

This command is used to retrieve the factory defined MAC address of the device, and its currently configured MAC address.

Table 2762 - ICMD_QUERY_VIRTUAL_PARAMETERS Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_parameters_en																																00h
				pcie_index				pf_index																		04h						

Table 2762 - ICMD_QUERY_VIRTUAL_PARAMETERS Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_parameters_context																																10h-FCh

Table 2763 - ICMD_QUERY_VIRTUAL_PARAMETERS Output Mailbox Field Description

Offset	Bits	Name	Description
00h	31:0	virtual_parameters_en	Each bit in this bitmask indicates whether the appropriate field in <i>virtual_parameters_context</i> is valid. 0x0: <i>virtual_link_down</i> Other bits are reserved
04h	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register
10h-FCh	1920	virtual_parameters_context	Current Virtual parameters to be modified. See Table 2766, “Virtual Parameters Context Layout,” on page 2292

30.5.6 ICMD_SET_VIRTUAL_PARAMETERS - Set Device Virtual Parameters

This command is used to override different configurations of the device

Table 2764 - ICMD_SET_VIRTUAL_PARAMETERS Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_parameters_en																																00h
				pcie_index				pf_index																								04h
virtual_parameters_context																																10h-FCh

Table 2765 - ICMD_SET_VIRTUAL_MAC Input Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	31:0	virtual_parameters_en	Each bit in this bitmask indicates whether the appropriate field in <i>virtual_parameters_context</i> should be modified. 0x0: <i>virtual_link_down</i> Other bits are reserved
04h	27:24	pcie_index	Internal domain index. valid only for Socket Direct devices, according to MPIR register
	23:20	pf_index	PCI Physical Function index. valid only for multi-PF device, according to MPEIN register
10h-FCh	1920	virtual_parameters_-context	Virtual parameters to be modified. See Table 2766, “Virtual Parameters Context Layout,” on page 2292

Table 2766 - Virtual Parameters Context Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
virtual_link_down																																00h
																																04h-ECh

Table 2767 - Virtual Parameters Context Field Descriptions

Offset	Bits	Name	Description
00h	31	virtual_link_down	When set, the device will logically emulate link state down for all associated functions.

30.6 Debug Commands

30.6.1 ICMD_SET_DIAGNOSTIC_PARAMS - Set Diagnostic Parameters

ICMD_SET_DIAGNOSTIC_PARAMS command configures the device to start/stop collecting diagnostics counters. In addition, it sets the counters list and the sampling period.

The counters report is triggered using the ICMD_QUERY_DIAGNOSTIC_COUNTERS command.

Note: *tracer_dump* must be set.

Table 2768 - ICMD_SET_DIAGNOSTIC_PARAMS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-04h
diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)																																08h-...

Table 2769 - ICMD_SET_DIAGNOSTIC_PARAMS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h-...	224	diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)	Diagnostic parameters context. See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223	

30.6.2 ICMD_QUERY_DIAGNOSTIC_PARAMS - Query Diagnostic Parameters

ICMD_QUERY_DIAGNOSTIC_PARAMS command queries for device configurations of collecting diagnostics counters.

Table 2770 - ICMD_QUERY_DIAGNOSTIC_PARAMS Output Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-04h
diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)																																08h-...

Table 2771 - ICMD_QUERY_DIAGNOSTIC_PARAMS Output Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h-...	224	diagnostic_params_context (See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223)	Diagnostic parameters context. See Table 2644, “DIAGNOSTIC_PARAMS_CONTEXT Input Structure Layout,” on page 2223	

30.6.3 ICMD_QUERY_DIAGNOSTIC_COUNTERS - Query Diagnostic Counters

ICMD_QUERY_DIAGNOSTIC_COUNTERS command reports to the tracer buffer the counters samples with the sampling time stamp. The sample buffer is cyclic, and when working in repetitive sampling mode, it might be overwritten if not queried often.

Table 2772 - ICMD_QUERY_DIAGNOSTIC_COUNTERS Input Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h-04h
num_of_samples																sample_index																08h
																																0Ch

Table 2773 - ICMD_QUERY_DIAGNOSTIC_COUNTERS Input Structure Field Descriptions

Offset	Bits	Name	Description	Access
08h	31:16	num_of_samples	The number of samples to return. Device might return up to the configured value HCA_CAP.log_number_of_samples. (See Section 29.3.4.10, “Debug Capabilities,” on page 1777). Note that the device can perform roll over when reaching number_of_samples. For example: if the user asks for 256 samples starting from index 128, the result will be: 128,...,255,0,...,127.	
	15:0	sample_index	The sample_index is the first sample index. The sample index shall be in the range of 0... 2 ^{HCA_CAP.log_number_of_samples} . See Section 29.3.4.10, “Debug Capabilities,” on page 1777.	

30.7 HP Commands (HP)

[Internal] These commands are HP only

30.7.1 ICMD_INIT_OCSD

This command initializes the OCSD (Option Card Sensor Data) reporting. The command passes a pointer to a table initialized by the BIOS/BMC and the device registers itself in it. It then periodically updates its sensor data (e.g. temperature).

Table 2774 - ICMD_INIT_OCSD Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																00h
address_hi																																04h
address_lo																																08h
																																0Ch-0FCh

Table 2775 - ICMD_INIT_OCSD Input Mailbox Field Description

Offset	Bits	Name	Description
04h	31:0	address_hi	MSB of OCSD registration table address.
08h	31:0	address_lo	LSB of OCSD registration table address.

30.7.2 ICMD_OCBB_INIT - Initialize OCBB report

This command is used by the option ROM to pass the location of the OCBB memory to the option card. Passing the pointer also enables the device support for OCBB functionality which is disabled by default.

Table 2776 - ICMD_OCBB_INIT Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
address_hi																																00h
address_lo																																04h

Table 2776 - ICMD_OCBB_INIT Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
																																08h-0FCh

Table 2777 - ICMD_OCBB_INIT Input Mailbox Field Description

Offset	Bits	Name	Description
00h	31:0	address_hi	MSB of OCBB buffer address.
04h	31:0	address_lo	LSB of OCBB buffer address.

30.7.3 ICMD_OCBB_QUERY_HEADER_STATUS - Get OCBB header initialization status

This command allows the option ROM to know when the OCBB header is updated by the option card, and that the option card is correctly registered to use the OCBB memory. The option ROM should not try to use “ICMD_OCBB_QUERY_ETOC_STATUS - Get OCBB ETOC initialization status” command before it verifies that the OCBB header is correctly updated by the option card.

Table 2778 - ICMD_OCBB_QUERY_HEADER_STATUS Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset	
status																																abs support	00h
																																	04h-0FCh

Table 2779 - ICMD_OCBB_QUERY_HEADER_STATUS Output Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	Status of the OCBB header 0: ready 1: busy 2: failed - not enough buffer space 3: failed - maximal instance cards reached Other values are reserved for future failure conditions.
	0	ahs_support	When set, the device uses AHS status reporting, so OCBB buffer memory is not required but the events should still be provided.

30.7.4 ICMD_OCBB_QUERY_ETOC_STATUS - Get OCBB ETOC initialization status

This command allows the option ROM to know when the OCBB ETOC is initialized. The option ROM should not try to use “ICMD_OCBB_SET_EVENT - Set OCBB event information” command before it verifies that the ETOC initialization is complete and successful.

Table 2780 - ICMD_OCBB_QUERY_ETOC_STATUS Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
status																																00h
																																04h-0FCh

Table 2781 - ICMD OCBB QUERY ETOC STATUS Output Mailbox Field Descriptions

Offset	Bits	Name	Description
00h	31:24	status	Status of the OCBB ETOC 0: ready 1: busy Other values are reserved for future failure conditions.

30.7.5 ICMD_OCBB_SET_EVENT - Set OCBB event information

This command is used by the option ROM to write an OCBB event information into the OCBB memory.

If the written event is unsupported (not included in the initialized ETOC list), an error response code is returned, and there will be no change made to the OCBB memory. This command allows updating events values at arbitrary size, but only allows for maximum 8 Bytes of information to be written at once. The Byte order of writing the event information segments is mandatory. It is expected that an option ROM will always update ALL the data in a defined event before writing data of a different event.

When the last byte of the event is provided, the device shall update the OCBB buffer with the whole event. Writing the last byte of the event is considered to be COMMITTING and the event is expected to be updated in OCBB memory following this write operation.

Event length should be provided with any fraction of the event data. The device uses the event length value to identify the end of the event data. When the event length is not an integer multiple of 8 bytes, the unused bytes should be set to 0 by the option ROM.

Table 2782 - ICMD_OCBB_SET_EVENT Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
event_data																																00h-04h
data_start_offset								data_length								event_length								event_number								08h
																																0Ch-0FCh

Table 2783 - ICMD_OCBB_SET_EVENT Input Mailbox Field Descriptions

Offset	Bits	Name	Description
00h-04h	64	event_data	
08h	31:24	data_start_offset	
	23:16	data_length	
	15:8	event_length	
	7:0	event_number	

30.8 Internal Commands

30.8.1 ICMD_GET_FRC_CHALLENGE - Get Factory Re-customization Token

[Internal] Internal Only

Table 2784 - ICMD_GET_FRC_CHALLENGE Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
frc_keypair_uuid[0]																																00h-0Cch
frc_keypair_uuid[1]																																10h-1Cch
frc_keypair_uuid[2]																																20h-2Cch
frc_keypair_uuid[3]																																30h-3Cch
frc_keypair_uuid[4]																																40h-4Cch
frc_keypair_uuid[5]																																50h-5Cch
frc_keypair_uuid[6]																																60h-6Cch
frc_keypair_uuid[7]																																70h-7Cch
frc_keypair_uuid[8]																																80h-8Cch
frc_keypair_uuid[9]																																90h-9Cch
frc_keypair_uuid[10]																																A0h-ACch
frc_keypair_uuid[11]																																B0h-BCch
frc_keypair_uuid[12]																																C0h-CCch

Table 2784 - ICMD_GET_FRC_CHALLENGE Input Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
frc_keypair_uuid[13]																																D0h-DCh
frc_keypair_uuid[14]																																E0h-ECh
frc_keypair_uuid[15]																																F0h-FCh

Table 2785 - ICMD_GET_FRC_CHALLENGE Input Mailbox Field Description

Offset	Bits	Name	Description	Access
10h*i - 10*i + 0Ch	128	frc_keypair_uuid[i]	Available FRC key-pair UUID[i]. Value 0x0 indicates an invalid entry	

Table 2786 - ICMD_GET_FRC_CHALLENGE Output Mailbox Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
frc_keypair_uuid																																00h-0Ch
version																																10h
time_to_live																																14h
base_mac																																18h-1Ch
psid																																20h-2Ch
firmware_version																																30h
nonce																																34h-7Ch
																																80h-FCh

Table 2787 - ICMD_GET_FRC_CHALLENGE Output Mailbox Field Description

Offset	Bits	Name	Description	Access
00h-0Ch	128	frc_keypair_uuid	The UUID of the key used to generate the challenge, and should be used to create the response.	
10h	31:24	version	Version of the challenge format	
14h	31:0	time_to_live	Expiration period for the Challenge	
18h-1Ch	64	base_mac	Device base MAC address, see Table 35 - MAC Address Layout	
20h-2Ch	128	psid	Device PSID, see <i>vsd_contd_psid</i> field in Table 766 - QUERY_-ADAPTER Parameters Block Field Descriptions.	
30h	31:0	firmware_version	Device Firmware version	
34h-7Ch	608	nonce	Random generated field for Challenge uniqueness.	

30.8.2 ICMD_SET_FRC - Set Factory Re-customization

[Internal] Internal Only