Broadcom Ethernet Network Adapter User Guide

User Guide
Version 235

Table of Contents

Introducing Ethernet Network Adapters....8
Broadcom Ethernet Network Adapters....8
Features of Broadcom Ethernet Network Adapters....9
Software and Hardware Features for Ethernet Network Adapters....9
Virtualization Features for Ethernet Network Adapters....12
VXLAN Feature for Ethernet Network Adapters....13
NVGRE/GRE/IP-in-IP/Geneve for Ethernet Network Adapters....14
Stateless Offloads Feature for Ethernet Network Adapters....14
Priority Flow Control Feature in Ethernet Network Adapters....15
Virtualization Offload Feature for Ethernet Network Adapters....16
SR-IOV Feature for Ethernet Network Adapters....19
Network Partitioning Feature for Ethernet Network Adapters....20
Security Features for Ethernet Network Adapters....20
RDMA over Converged Ethernet Feature in Ethernet Network Adapters....20
Enhanced Network Stack (ENS)....23
XDP and AF_XDP....25
Supported Feature Combinations for Ethernet Network Adapters....26
Unsupported Features for Ethernet Network Adapters....26
Installing Broadcom Ethernet Network Adapter Hardware....27
Broadcom Ethernet Network Adapter Safety Precautions....27
Hardware Requirements for Broadcom Ethernet Network Adapters....27
Replacing the Ethernet Network Adapter Mounting Brackets....28
Installing the Ethernet Network Adapter....28
Cable/Interconnect Compatibility List for Broadcom Ethernet Network Adapters....28
Cable/Interconnect Compatibility for BCM9575XX, BCM9574XX, and Earlier Adapters....29
Cable/Interconnect Compatibility for BCM957608 (Thor 2) Adapters....33
UEFI HII Menu for Ethernet Network Adapters....34
Installing Software for Ethernet Network Adapters....49
Supported Operating Systems for Ethernet Network Adapters....49
Installing the Linux Driver on Ethernet Network Adapters....50
Installing the L2 and RoCE Drivers Automatically....50
Installing and Configuring the Software Manually....53
Useful Linux Commands....63
Installing the VMware Driver on Ethernet Network Adapters....64
Installing the Windows Driver on Ethernet Network Adapters....65
Advanced Properties for Ethernet Network Adapters....65

Windows Driver RoCE Configuration....67
Windows Server Behavior....67
Updating the Firmware on Ethernet Network Adapters....68
Updating the Firmware Online (Windows/Linux)....69
Updating the Firmware with the Automated Installer for Linux....72
Updating the Firmware Manually on Linux/ESX....72
Updating the Firmware on Windows....74
Verifying the Firmware....75
Configuring Ethernet Network Adapters....78
Tunneling Examples for Ethernet Network Adapters....78
Link Aggregation on Ethernet Network Adapters....81
Configuring a RoCE Link Aggregation Group....81
Ethernet Physical Link Control Policy....86
Configuring Auto-Negotiation on Ethernet Network Adapters....87
Configuring Port Breakout on BCM95750X/BCM957608 Ethernet Adapters....92
Configuring 200G Link Speeds on 2 x 100G BCM957508 Ethernet Network Adapters....94
Configuring 200G Link Speeds on 2 x 100G BCM957608 Ethernet Network Adapters....98
Configuring 200G and 400G Link Speeds on 2 x 200G BCM957608 Dual-Port Ethernet Network Adapters....102
PXE Boot on Ethernet Network Adapters....108
Configuring DHCP for PXE/iPXE....114
Configuring TFTP....117
Configuring HTTP/HTTPS....118
SR-IOV and Use Case Examples for Ethernet Network Adapters....119
RDMA SR-IOV for Ethernet Network Adapters....121
NPAR and Use Case Examples for Ethernet Network Adapters....122
DCBX – Data Center Bridging....127
Configuring Peer Memory Direct with BCM95750X/BCM957608 Network Adapters....130
Windows Configuration Information....132
ESXi Configuration Information....132
Precision Time Protocol....132
PTP Specification....133
PTP Delay Measurements....134
Installing PTP....136
Configuring PTP....137
ptp4l Configuration File....139
Tuning Ethernet Network Adapters....140
NIC Tune....140
Performance Profiles....142
BIOS Tuning on Ethernet Network Adapters....142

NPS (NUMA Per Socket)....143
X2APIC....143
Determinism Control and Determinism Slider....144
APBDIS....145
Preferred I/O and Enhanced Preferred I/O....146
PCIe Ten Bit Tag....147
Memory Clock Speed....148
L3 LLC (Last Level Cache) as NUMA....148
Socket/Inter-Chip Global Memory Interconnect (xGMI)....149
TCP Performance Tuning on Ethernet Network Adapters....150
TCP BIOS Performance Tuning....150
Adapter Tuning....151
NUMA: Local vs. Non Local....152
Configuring Queues....153
Configuring IRQ and Application Affinity....153
TX and RX Flow Steering....157
TX and RX Queue Size....157
Interrupt Moderation....158
GRO (Generic Receive Offload)....158
Relaxed Ordering....158
PCIe MRRS (Maximum Read Request Size)....159
Operating System Tuning (Linux)....159
TCP Example with the BCM957508-P2100G....160
Configuring RSS for Performance in Windows....162
Setting Up RSS....163
Monitoring Receive Queues/Cores....166
Expanding to New Counters....169
Configuring RSSv2....171
DPDK Tuning for Ethernet Network Adapters....172
TCP BIOS Performance Tuning....172
Kernel Tuning....173
Adapter Tuning....173
Application Tuning....174
DPDK Results....174
IP Forwarding Tunings for Ethernet Network Adapters....175
BIOS Tuning....175
Kernel Tuning....175
Adapter Tuning....176
IP Forwarding Results....177
Gathering Statistics for Ethernet Network Adapters....178

Ethernet Statistics **178**
Linux Statistics 178
Windows Statistics 179
Performance Counters for Windows **182**
Ethtool Counters **185**
Ethernet Network Adapter Utilities **206**
NICCLI **206**
Installing the NICCLI Configuration Utility 206
Using the NICCLI Configuration Utility 208
NICCLI Logging 210
NICCLI Command Line Options 211
NICCLI Configuration Utility Commands 212
fw Command 213
nvm Command 215
qos Command 217
linkdiag Command 221
serdes Command 225
cable Command 227
link Command 229
timesync Command 231
counters Command 233
vf Command 234
tunnel Command 235
msix Command 236
mh Command 237
resmgmt Command 238
ccparams Command 239
debug Command 240
show Command 241
pcie Command 242
VMWare Signed and Unsigned Command Mapping 243
Mapping NICCLI Commands (233 to 234 and Later Versions) 248
NIC Apps **254**
System Status Check Utility 255
NIC Tune **255**
RoCE Analyzer **257**
Tool Command 258
Software Command 258
Performance Command 259
BIOS Command 260

Layer2 Connectivity Command....260
Link Command....261
NVM Command....262
UDCC Command....262
Congestion Control Commands....263
QoS Command....263
Recommendations Command....264
Debug Telemetry Command....265
Statistics Command....265
All Command....266
SOS Reporting Tool....266
RDMA over Converged Ethernet (RoCE)....269
Configuring RoCE on Linux....270
Modifying RoCE NVM Configuration Using NICCLI....272
Modifying QoS Configuration Using NICCLI....273
Modifying RoCE Configuration Using bnxtsetupcc.sh....274
DSCP and VLAN....274
Bandwidth Allocation....275
Advanced RDMA over Converged Ethernet (RoCE) Network Configuration....275
RoCE Congestion Control....276
Congestion Control Tuning Parameters....279
Quality of Service....282
Validating RDMA over Converged Ethernet (RoCE) Network on Linux....284
Configuring RoCE on VMware....285
Configuring RoCE on Windows....287
Tuning RoCE for Ethernet Network Adapters....295
Disable CPU Power Saving....295
Tuning Applications for High QP Count....295
Improving RDMA Workload Performance....296
Switch Configuration for RDMA over Converged Ethernet (RoCE)....296
Supporting Multiple Lossless Queues and Traffic Classes....304
Retrieving RoCE Statistics....306
RoCE Counter Definitions....310
Example RoCE + TCP Configuration on Ethernet Network Adapters....315
Windows Configuration Information....318
ESXi Configuration Information....319
TruFlow....320
System Configuration....321
Enabling TruFlow....321

Configuring the Network on the Host....322
Creating a VM....323
Configuring Networking on the VM....324
Testing TruFlow....325
Sample Logs....327
Supported Patterns and Actions....334
Conclusion....335
OVS and TC and OVS-DPDK Offload....336
OVS-TC and OVS-DPDK Feature List....336
NVM Changes....342
Installing and Using the OVS-TC and OVS-DPDK....342
Configuring OVS-TC and OVS-DPDK....345
OVS Flow Offloads....355
Test Guidance....357
Hardware and Regulatory Information....366
Ethernet Network Adapter Functional Descriptions....366
Ethernet Network Adapter Network Link and Activity LEDs....392
Ethernet Network Adapter Regulatory and Safety Approvals....413
Frequently Asked Questions for Ethernet Network Adapters....415
Other Broadcom Resources....424
Documentation Legal Notice....425

Introducing Ethernet Network Adapters

Provides introductory information to quickly install and configure Broadcom Ethernet network adapters.

Designed for today's enterprise and cloud-scale environments, Broadcom's Ethernet network adapters are the ideal solution for high-performance virtualization, intelligent flow processing, secure data center connectivity, and machine learning.

The introduction consists of the following sections:

- Broadcom Ethernet Network Adapters
- Features of Broadcom Ethernet Network Adapters

Broadcom Ethernet Network Adapters

Provides a list of available Broadcom Ethernet network adapters.

The Broadcom BCM95719, BCM95720, BCM9574XX, BCM95750X, and BCM957608 family of Ethernet network adapters are highly-integrated, full-featured Ethernet LAN controllers optimized for data center and cloud infrastructures. These controllers support 400G/200G/100G/50G/40G/25G/10G/1G in single, dual, or quad-port configurations. These controllers can support up to sixteen lanes of PCIe Gen3. The BCM95750X family additionally supports PCIe Gen4 and the BCM57608 supports PCIe Gen5. An extensive set of stateless offloads and virtualization offloads to enhance packet processing efficiency is included to enable low-overhead, high-speed network communications.

The following table shows the list of Broadcom channel cards. For functional descriptions and figures for each card, see Ethernet Network Adapter Functional Descriptions. For the latest drivers and software images, select the appropriate card from https://www.broadcom.com/products/ethernet-connectivity/network-adapters and then the **Downloads** tab of the card.

Table 1: Broadcom Network Adapter List

ASIC	Part Number	Form Factor	Connector	Port Number and Speed
BCM5719	BCM95719A1904AC	PCIe	1GBASE-T	4 x 1GbE
BCM5719	BCM95719N1905C	OCP 3.0	1GBASE-T	4 x 1GbE
BCM5720	BCM95720A2003AC	PCIe	1GBASE-T	2 x 1GbE
BCM57412	BCM957412A4120AC	PCIe	SFP+	2 x 10GbE
BCM57412	BCM957412N4120C	OCP 3.0	SFP+	2 x 10GbE
BCM57414	BCM957414A4142CC	PCIe	SFP28	2 x 25/10GbE
BCM57414	BCM957414N4140C	OCP 3.0	SFP28	2 x 25/10GbE
BCM57416	BCM957416A4160C	PCIe	RJ-45	2 x 10GBASE-T
BCM57416	BCM957416N4160C	OCP 3.0	RJ-45	2 x 10GBASE-T
BCM57504	BCM957504-P425G	PCIe	SFP28	4 x 25/10GbE
BCM57504	BCM957504-N425G	OCP 3.0	SFP28	4 x 25/10GbE
BCM57504	BCM957504-N1100G	OCP 3.0	QSFP56	1 x 100GbE
BCM57508	BCM957508-P2100G	PCIe	QSFP56	2 x 100GbE
BCM57508	BCM957508-N2100G	OCP 3.0	QSFP56	2 x 100GbE

ASIC	Part Number	Form Factor	Connector	Port Number and Speed
BCM57608	BCM957608-P2200GQF00	PCIe	QSFP112	2 x 200GbE
BCM57608	BCM957608-N2200GQP00	OCP 3.0	QSFP112	2 x 200GbE
BCM57608	BCM957608-P1400GDF00	PCIe	QSFP112-DD	1 x 400GbE
BCM57608	BCM957608-N1400GDP00	OCP 3.0	QSFP112-DD	1 x 400GbE

Features of Broadcom Ethernet Network Adapters

Provides descriptions of the features available in Ethernet network adapters.

See the following sections for device features:

- Software and Hardware Features for Ethernet Network Adapters
- Virtualization Features for Ethernet Network Adapters
- VXLAN Feature for Ethernet Network Adapters
- NVGRE/GRE/IP-in-IP/Geneve Features for Ethernet Network Adapters
- Stateless Offloads Feature for Ethernet Network Adapters
- Priority Flow Control Feature for Ethernet Network Adapters
- Virtualization Offload Feature for Ethernet Network Adapters
- SR-IOV Feature for Ethernet Network Adapters
- Network Partitioning (NPAR) Feature for Ethernet Network Adapters
- Security Feature for Ethernet Network Adapters
- RDMA over Converged Ethernet – RoCE Feature for Ethernet Network Adapters
- Enhanced Network Stack (ENS)
- XDP and AF_XDP
- Supported Feature Combinations for Ethernet Network Adapters
- Unsupported Feature Combinations for Ethernet Network Adapters

Software and Hardware Features for Ethernet Network Adapters

Describes the included software and hardware features for Ethernet network adapters.

The following table provides a list of software and hardware features for Ethernet network adapters.

Table 2: Host Interface Features

Feature	BCM95741X	BCM95750X	BCM957608
Host Interface	PCIe Gen3 Gen3: 8 Gb/s Gen2: 5 Gb/s Gen1: 2.5 Gb/s	PCIe Gen4 Gen4: 16.0 Gb/s Gen3: 8 Gb/s Gen2: 5 Gb/s Gen1: 2.5 Gb/s	PCIe Gen5 Gen5: 32.0 Gb/s Gen4: 16.0 Gb/s Gen3: 8 Gb/s Gen2: 5 Gb/s Gen1: 2.5 Gb/s
Number of PCIe Lanes	PCIe Edge Connectors: • x8 **Note:** N series adapters support OCP 3.0 interfaces.	PCIe Edge Connectors: • x16 **Note:** N series adapters support OCP 3.0 interfaces.	PCIe Edge Connectors: • x16 **Note:** N series adapters support OCP 3.0 interfaces.
Vital Product Data (VPD)	Supported	Supported	Supported
Alternate Routing ID (ARI)	Supported	Supported	Supported
Function Level Reset (FLR)	Supported	Supported	Supported
Advanced Error Reporting	Supported	Supported	Supported
PCIe ECNs	Support for TLP Processing Hints (TPH), Latency Tolerance Reporting (LTR), and Optimized Buffer Flush/Fill (OBFF).	Support for TLP Processing Hints (TPH), Latency Tolerance Reporting (LTR), and Optimized Buffer Flush/Fill (OBFF).	Support for TLP Processing Hints (TPH), Latency Tolerance Reporting (LTR), and Optimized Buffer Flush/Fill (OBFF).
MSI-X Interrupt Vector per Queue	1 per RSS queue, 1 per NetQueue, 1 per Virtual Machine Queue (VMQ).	1 per RSS queue, 1 per NetQueue, 1 per Virtual Machine Queue (VMQ).	1 per RSS queue, 1 per NetQueue, 1 per Virtual Machine Queue (VMQ).
IP Checksum Offload	Support for transmit and receive side.	Support for transmit and receive side.	Support for transmit and receive side.
TCP Checksum Offload	Support for transmit and receive side.	Support for transmit and receive side.	Support for transmit and receive side.
UDP Checksum Offload	Support for transmit and receive side.	Support for transmit and receive side.	Support for transmit and receive side.
NDIS TCP Large Send Offload	Support for LSOV1 and LSOV2.	Support for LSOV1 and LSOV2.	Support for LSOV1 and LSOV2.
NDIS Receive Segment Coalescing (RSC)	Support for Windows environments.	Support for Windows environments.	Not Supported.
TCP Segmentation Offload (TSO)	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux environments only.
Large Receive Offload (LRO)	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux environments only.
Generic Receive Offload (GRO)	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux and VMware environments.	Supports hardware acceleration in Linux environments only.
Receive Side Scaling (RSS)	Supports RSS in Windows, Linux, and VMware environments.	Supports RSS in Windows, Linux, and VMware environments.	Supports RSS in Linux environments only.

Feature	BCM95741X	BCM95750X	BCM957608
Header-Payload Split	Enables the software TCP/IP stack to receive TCP/IP packets with header and payload data split into separate buffers. Supported in Windows, Linux, and VMware environments.	Enables the software TCP/IP stack to receive TCP/IP packets with header and payload data split into separate buffers. Supported in Windows, Linux, and VMware environments.	Enables the software TCP/IP stack to receive TCP/IP packets with header and payload data split into separate buffers. The BCM57608 supports Linux environments only.
Accelerated Receive Flow Steering (aRFS)	Hardware acceleration support for Linux.	Hardware acceleration support for Linux.	Hardware acceleration support for Linux.
Jumbo Frames	Supported	Supported	Supported
iSCSI Boot	Supported through UEFI iSCSI stack. Legacy iSCSI boot not supported.	Supported through UEFI iSCSI stack. Legacy iSCSI boot not supported.	Supported through UEFI iSCSI stack. Legacy iSCSI boot not supported.
NIC Partitioning (NPAR)	Supports up to eight Physical Functions (PFs) per port, or up to 16 PFs per silicon. This option is configurable in NVRAM.	Supports up to eight Physical Functions (PFs) per port, or up to 16 PFs per silicon. This option is configurable in NVRAM.	Supports up to eight Physical Functions (PFs) per port, or up to 16 PFs per silicon. This option is configurable in NVRAM.
RDMA over Converged Ethernet (RoCE)	Supports RoCE v2 for Windows, Linux, and VMware.	Supports RoCE v2 for Windows, Linux, and VMware.	Supports RoCE v2 for Linux.
Data Center Bridging (DCB)	Supports DCBX (IEEE and CEE (Windows only) specification), PFC, and AVB.	Supports DCBX (IEEE and CEE (Windows only) specification), PFC, and AVB.	Supports DCBX (IEEE specification), PFC, and AVB.
Network Controller Sideband Interface (NC-SI)	Supported	Supported	Supported
Wake on LAN (WOL)	Supported in only OCP designs.	Supported in only OCP designs.	Not Supported.
PXE Boot	Supported	Supported	Supported (UEFI only).
UEFI Boot	Supported	Supported	Supported
Pause Flow Control (IEEE 802.3x)	Supported	Supported	Supported
Priority Flow Control (IEEE 802.1Qbb)	Supported	Supported	Supported
Auto-negotiation	Supported	Supported	Supported
IEEE 802.1q VLAN	Supported	Supported	Supported
Interrupt Aggregation	Supported	Supported	Supported
MAC/VLAN Filters	Supported	Supported	Supported
PTP	Supported only in Linux.	Supported only in Linux.	Supported only in Linux.
eDPC	Supported only in Linux.	Supported only in Linux.	Not Supported.

Note: RoCE v1 support is deprecated on all devices. RoCE v1 is not removed from the code and displays in the `ibv_devinfo` command output. This was done intentionally to keep supporting the same GID numbering for RoCE v2.

Virtualization Features for Ethernet Network Adapters

Describes the supported virtualization features for Ethernet network adapters.

The following table lists the virtualization features of Broadcom Ethernet network adapters.

Table 3: Virtualization Features

Feature	BCM95741X	BCM95750X	BCM957608
Linux KVM Multiqueue	Supported	Supported	Not Supported
VMware NetQueue	Supported	Supported	Not Supported
NDIS Virtual Machine Queue (VMQ)	Supported	Supported	Not Supported
Virtual eXtensible LAN (VXLAN) – Aware stateless offloads (IP/UDP/TCP checksum offloads, VLAN insertion/removal, NetQueue, VMQ, RSS, TCP segmentation offload, large send offload, generic receive offload)	Supported	Supported	Supported
Generic Routing Encapsulation (GRE) – Aware stateless offloads (IP/UDP/TCP checksum offloads, VLAN insertion/removal, VMQ, RSS, TCP segmentation offload, generic receive offload)	Supported	Supported	Supported
Network Virtualization using Generic Routing Encapsulation (NVGRE) – Aware stateless offloads (IP/UDP/TCP checksum offloads, VLAN insertion/removal, VMQ, RSS, large send offload)	Supported	Supported	Supported
Generic Network Virtualization Encapsulation (Geneve) – Aware Stateless offloads (IP/UDP/TCP checksum offloads, VLAN insertion/removal, NetQueue, RSS, TCP segmentation offload, generic receive offload)	Supported	Supported	Supported
IP-in-IP aware stateless offloads (IP/UDP/TCP checksum offloads, VLAN insertion/removal, NetQueue, RSS, TCP segmentation offload, generic receive offload)	Supported	Supported	Supported

Feature	BCM95741X	BCM95750X	BCM957608
SR-IOV v1.0	128 Virtual Functions (VFs) for Guest Operating Systems (GOS) per device.	128 VFs for GOS per device.	128 VFs for GOS per device.
Edge Virtual Bridging (EVB) (IEEE 802.1Qbg) Edge Virtual Bridging (EVB) enables switching of traffic between PFs/VFs, forwarding of outgoing network traffic from PFs/VFs to appropriate network ports, and steering of incoming network traffic to appropriate PFs/VFs. Both VEB (local switching in the adapter) and VEPA (switching in the adjacent switch) EVB modes of operation are supported.	The EVB features supported are: • 128 VFs and up to 128 queues per VF (flexible allocation across PFs/VFs). • PCIe AER, TPH, FLR support. • Virtual Ethernet Bridge (VEB)/Virtual Ethernet Port Aggregator (VEPA). • MAC/VLAN filtering. • VF isolation and source pruning. • Stateless and packet steering offloads per VF. • Forwarding of unicast frames based on {Tunnel ID (optional), Destination MAC, and VLAN ID (optional)}. • Frame replication for multicast, broadcast, and promiscuous mode. • Source pruning – Provide support for source knockout (prevent sending a multicast or broadcast frame back to the source). • Mirroring of traffic to a specific PF or VF. • Packet editing – VLAN insert/swap/delete. • Anti-spoof checks.	The EVB features supported are: • 128 VFs and up to 128 queues per VF (flexible allocation across PFs/VFs). • PCIe AER, TPH, FLR support. • Virtual Ethernet Bridge (VEB)/Virtual Ethernet Port Aggregator (VEPA). • MAC/VLAN filtering. • VF isolation and source pruning. • Stateless and packet steering offloads per VF. • Forwarding of unicast frames based on {Tunnel ID (optional), Destination MAC, and VLAN ID (optional)}. • Frame replication for multicast, broadcast, and promiscuous mode. • Source pruning – Provide support for source knockout (prevent sending a multicast or broadcast frame back to the source). • Mirroring of traffic to a specific PF or VF. • Packet editing – VLAN insert/swap/delete. • Anti-spoof checks.	The EVB features supported are: • 128 VFs and up to 128 queues per VF (flexible allocation across PFs/VFs). • PCIe AER, TPH, FLR support. • Virtual Ethernet Bridge (VEB)/Virtual Ethernet Port Aggregator (VEPA). • MAC/VLAN filtering. • VF isolation and source pruning. • Forwarding of unicast frames based on {Tunnel ID (optional), Destination MAC, and VLAN ID (optional)}. • Frame replication for multicast, broadcast, and promiscuous mode. • Source pruning – Provide support for source knockout (prevent sending a multicast or broadcast frame back to the source).
MSI-X vector port	74 per port default value (two-port configuration). 16 (8 for BCM9575XX devices) per VF and is configurable in HII.	74 per port default value (two-port configuration). 8 per VF and is configurable in HII.	512 entries per PF in non-NPAR cases.

VXLAN Feature for Ethernet Network Adapters

Describes the VXLAN feature for Ethernet network adapters.

A Virtual eXtensible Local Area Network (VXLAN), defined in IETF RFC 7348, addresses the need for overlay networks within virtualized data centers accommodating multiple tenants. VXLAN is a Layer 2 overlay or tunneling scheme over a Layer 3 network. Only VMs within the same VXLAN segment can communicate with each other.

NVGRE/GRE/IP-in-IP/Geneve for Ethernet Network Adapters

Describes the NVGRE/IP-in-IP/Geneve features for Ethernet network adapters.

Network Virtualization using GRE (NVGRE) is similar to a VXLAN. See IETF RFC 7637 for additional information.

Note: Checksum offload must be enabled when using NVGRE.

Stateless Offloads Feature for Ethernet Network Adapters

Describes the Stateless Offloads feature for Ethernet network adapters.

This section contains the following information on stateless offloads:

- IP, TCP, UDP Checksum Offload
- UDP Fragmentation Offload
- TCP Segmentation Offload and Large Send Offload
- Generic Receive Offload (GRO) and Large Receive Offload (LRO)
- Header and Data Split
- VLAN Tag Insertion and Removal
- Packet Steering
- Data Center Bridging

IP, TCP, UDP Checksum Offload

Host software can configure the Ethernet controller to calculate IP, TCP, and UDP checksums as described in RFC 791, RFC 793, and RFC 768, respectively. The first step in checksum calculation is determining the start of an IP and UDP datagram and TCP segment within a frame, which could vary depending on whether the frame is tagged (VLAN) or encapsulated with an LLC/SNAP header. Then the checksum is computed from the start to the end of the datagram and inserted into the appropriate location in the protocol header. The Ethernet controller is designed to support checksum calculation on all frame types and also on IP datagrams and TCP segments containing options.

UDP Fragmentation Offload

UDP Fragmentation Offload (UFO) is a feature that enables the software stack to offload fragmentation of large UDP/IP datagrams into multiple UDP/IP packets of size suitable for transmission. Enabling UFO can result in reduced CPU load for UDP applications. Support for this feature is only available in the Linux environment.

TCP Segmentation Offload and Large Segment Offload

Large Segment Offload (LSO) enables the software stack to offload the segmentation of large TCP messages into multiple TCP/IP packets of a size suitable for transmission. Enabling LSO can reduce the CPU load for TCP applications. This feature is also called TCP Segmentation Offload (TSO).

Generic Receive Offload (GRO) and Large Receive Offload (LRO)

Note: BCM5741X adapters do not support hardware LRO if Geneve traffic is present. BCM5750X and BCM957608 adapters support hardware LRO if Geneve traffic is present.

Generic Receive Offload (GRO) and Large Receive Offload (LRO) are hardware accelerations for TCP data reception. Both GRO and LRO modes of TCP receive offload are supported by the Ethernet Controller's Transparent Packet Aggregation (TPA) feature. Enabling GRO and LRO can significantly reduce CPU load and increase throughput for TCP applications by reducing the number of received messages, interrupts, and DMA operations. TPA aggregates TCP streams by managing context entries. Each entry in the TPA context is identified by the 4-tuple: source IP, destination

IP, source TCP port, and destination TCP port. GRO is the preferred TPA mode as packet boundaries are preserved for network routing applications, which may enable LSO for transmission.

Note: LRO is automatically disabled when the MTU is greater than 4096

Header and Data Split

Header-payload split is a feature that enables the software TCP/IP stack to receive TCP/IP packets with header and payload data split into separate buffers. Windows and Linux environments support this feature. The following are potential benefits of header-payload split:

- Enables compact and efficient caching of packet headers into host CPU caches. This caching can improve receive-side TCP/IP performance.
- Enables page flipping and zero-copy operations by the host TCP/IP stack, which can further improve the performance of the receive path.

VLAN Tag Insertion and Removal

On the TX path, the Ethernet controller can insert IEEE 802.1Q-compliant VLAN tags into transmitted frames and extract the VLAN tags from received frames. On the RX path, the Ethernet controller supports receiving VLAN-tagged (IEEE 802.1q-compliant) packets. If a function is configured to strip the VLAN tag, the VLAN tag is stripped from the IEEE 802.1q-compliant packet at reception and placed in a receive completion record.

Packet Steering

Receive Side Scaling (RSS)

RSS is a scalable networking technology that enables receive packet processing to be balanced across multiple processors in the system while maintaining in-order delivery of the data. RSS enables different packets, received by a single network adapter, to be processed on different CPUs/cores in parallel while preserving in-order delivery of TCP connections. RSS uses a Toeplitz algorithm, which uses a tuple hashing on the received frames and forwards it to a deterministic CPU for frame processing. This algorithm allows streamlined frame processing and balances CPU utilization. An indirection table maps the stream to a CPU. Symmetric RSS allows the mapping of packets of a given TCP or UDP flow to the same receive queue. BCM57608 supports XOR and checksum algorithms along with Toeplitz.

Accelerated Receive Flow Steering

Accelerated RFS (aRFS or RFS) is an Ethernet controller feature that improves packet reception efficiency by delivering packets to queues based on the CPU locality of the application. This feature reduces memory access latency and improves performance. Accelerated RFS takes precedence over RSS when enabled and configured. If the incoming flow does not match any existing n-tuple filters, it is steered according to the RSS hash.

Note: Ethtool N tuple filters that are created with port number 0 will not operate.

Note: Hash collision may occur during 5-tuple filter creation on BCM5741X Ethernet adapters and lead to filter creation failure.

Data Center Bridging

Data Center Bridging (DCB) is a set of protocols and capabilities (for example, DCBX, LLDP, ETS, and PFC) for use in a data center environment. Broadcom Ethernet network adapters' support for priority flow control is described in the Priority Flow Control section.

Priority Flow Control Feature in Ethernet Network Adapters

Describes the Priority Flow Control feature for Ethernet network adapters.

Priority Flow Control (PFC) is a standard-compliant backpressure mechanism implemented in Broadcom Ethernet network adapters. The goal of PFC is to backpressure congested priority traffic flow without affecting the traffic flows of

uncongested priorities and to ensure that packets are not dropped in burst or transient scenarios. PFC can be used in a network with real-time or time-sensitive traffic because it can provide differential treatment to traffic classes. For example, using PFC lower priority Internet traffic can be backpressured, leaving the higher priority traffic, like VOIP and streaming video, flowing through the link without flow control.

Virtualization Offload Feature for Ethernet Network Adapters

Describes the Virtualization Offload feature for Ethernet network adapters.

This section contains the following information on virtualization offload:

- Multiqueue Support
- KVM/Xen Multiqueue
- Virtual Machine Queue
- Tunneling Offload

Multiqueue Support

Broadcom Ethernet network adapters support Multiqueue in the hardware.

KVM Multiqueue (BCM5741X and BCM5750X Only)

Kernel Virtual Machine (KVM) Multiqueue processes the received packet's destination MAC address, IEEE 802.1Q VLAN tag, or both. After processing, VM/Multiqueue classifies the incoming frames and returns the frames to different queues of the host stack. The classification, combined with the ability to DMA the frames directly into a virtual machine's memory allows scaling of virtual machines across multiple processors.

Virtual Machine Queue (BCM5741X and BCM5750X Only)

The NDIS Virtual Machine Queue (VMQ) is a feature that Microsoft supports to improve Hyper-V network performance. The VMQ feature supports packet classification based on the destination MAC address to return received packets on different completion queues. This packet classification combined with the ability to DMA packets directly into a virtual machine's memory allows the scaling of virtual machines across multiple processors.

VMware NetQueue

The VMware NetQueue is a feature that is similar to Microsoft's NDIS VMQ feature. The NetQueue feature supports packet classification based on the destination MAC address and VLAN to return received packets on different NetQueues. This packet classification, combined with the ability to DMA packets directly into a virtual machine's memory allows the scaling of virtual machines across multiple processors.

Xen Multiqueue

Xen multiqueue enables network device drivers to dedicate each Rx queue to a specific guest operating system. This feature means the network device drivers can allocate physical memory from the set of memory pages assigned to a specific guest operating system.

Tunneling Offload

Stateless Transport Tunnel Offload (STT) is a tunnel encapsulation that enables overlay networks in virtualized data centers. STT uses IP-based encapsulation with a TCP-like header. No TCP connection state is associated with the tunnel and that is why STT is stateless. Open Virtual Switch (OVS) uses STT. An STT frame contains the STT frame header and payload. The payload of the STT frame is an untagged Ethernet frame. The STT frame header and encapsulated payload are treated as the TCP payload and TCP-like header. The IP header (IPv4 or IPv6) and Ethernet header are created for each transmitted STT segment. Broadcom Ethernet adapters support Network Overlays or Tunneling, specifically VXLAN, variants of GRE, and IP-in-IP. Both VXLAN and NVGRE support a larger scale than basic IEEE 802.1Q VLANs, using

a 24-bit label space rather than a 12-bit VID. Both are L2-in-L3 tunneling methods with NVGRE using GRE to carry the tunnel label and VXLAN using UDP to identify the tunnel label. Stateless offload for tunneling and encapsulated frames in the Broadcom Ethernet network adapter family applies to VXLAN, Geneve, L2GRE, NVGRE, and also IP-in-IP scheme. The following section describes VXLAN as an example to discuss the general support for this feature. The difference between different tunneling/encapsulation schemes is noted when it is applicable. All the offloads described in this section are supported by both physical and virtual functions.

VXLAN

A Virtual eXtensible Local Area Network (VXLAN), defined in IETF RFC 7348, addresses the need for overlay networks within virtualized data centers that accommodate multiple tenants. The VXLAN scheme and related protocols are defined in IETF RFC 7348. VXLAN is a Layer 2 overlay or tunneling scheme over a Layer 3 network. Each overlay is termed a VXLAN segment. Only VMs within the same VXLAN segment can communicate with each other. Each VXLAN segment is identified through a 24-bit segment ID called the VXLAN Network Identifier (VNI). This allows up to 16M VXLAN segments to coexist within the same administrative domain. The UDP destination port identifies the presence of a VXLAN tunnel.

GRE and NVGRE

Broadcom Ethernet adapters Generic Routing Encapsulation (GRE) per RFC 2784 and RFC 2890. Network Virtualization using GRE (NVGRE) is a Layer 2 overlay, used to address the need for subnets for overlay networks with larger numbers of VLANs. As with the VXLAN scheme, each NVGRE segment is scoped through a 24-bit identifier, called Virtual Subnet Identifier (VSID), in a GRE header to support up to 16M virtual network segments.

Geneve

Broadcom Ethernet adapters support Generic Network Virtualization Encapsulation (Geneve, also known as Next Generation Encapsulation). It leverages the same concepts as VXLAN, using a UDP destination port to identify the presence of the Geneve tunnel header. A primary goal for Geneve is to enable the transport of metadata (system state) from a source endpoint to one or more destination endpoints within the virtual network indicated by the tunnel identifier.

IP-in-IP

IP-in-IP is a Layer 3 overlay or tunneling scheme over a Layer 3 network. IP-in-IP is a method by which an IP datagram may be encapsulated (carried as payload) within another IP datagram to alter the routing of the inner IP packets and allow them to be delivered to an intermediate destination that would otherwise not be selected by the destination address field in the inner IP header. IP encapsulation with IP is defined in RFC 2003.

Checksum Offload

The following checksums are computed on transmit path and then computed and verified on the receive path.

- Outer IPv4 checksum if the outer IP datagram is an IPv4 datagram.
- Outer UDP checksum (if non-zero): The current VXLAN IETF draft suggests that the outer UDP checksum should be transmitted as zero. If the outer UDP checksum field in the VXLAN frame received is zero, then the frame is accepted without computing the outer UDP checksum. If the outer UDP checksum field in the VXLAN frame received is non-zero, the outer UDP checksum should be computed. (The IETF draft allows the receiver to ignore the outer UDP checksum when it is set to non-zero.) This item is not available in L2GRE/NVGRE/IP-in-IP aware offload.
- Inner IPv4 checksum if the inner IP datagram is an IPv4 datagram.
- Inner UDP or TCP checksum.

VLAN Tagging

A VXLAN frame supports the following tagging options:

- No IEEE 802.1Q tag in inner and outer datagrams.
- IEEE 802.1Q tag in the outer IP datagram only.
- IEEE 802.1Q tag in the inner IP datagram only.
- IEEE 802.1Q tags in both the inner and outer IP datagrams.

Note: The device supports insertion and removal of outer IEEE 801.Q Tag for VXLAN/GRE/IP-in-IP frames. It also supports the insertion and removal of inner IEEE 801.Q Tag for VXLAN.frames. The device retains the inner IEEE 801.Q Tag for NVGRE frames in the inner packet.

VMQ (BCM5741X and BCM5750X Only)

For VXLAN frames, the NetQueue uses the following fields for the queue selection:

- Inner destination MAC address.
- Outer destination MAC address.
- VXLAN Network Identifier (VNI).

The device supports NetQueue selection based on any combination of the previous fields.

Note: For GRE/IP-in-IP frames, the VMQ selection is performed using the Ethernet header of the encapsulated packet (inner packet) that includes the inner destination MAC address and inner 802.1Q Tag (optional).

RSS

For VXLAN frames, two options exist for RSS queue selection:

- RSS hash computation based on outer UDP/IP headers: The VXLAN IETF draft recommends that the source port is set based on the hash of inner headers. This setting allows the RSS hash computation based on outer UDP/IP headers as a viable option.
- RSS hash computation based on inner UDP/IP or TCP/IP headers: This option requires 2-tuple or 4-tuple hash computation based on inner headers. The inner header is parsed for RSS hash computation. The RSS hash computation is performed in parallel with other checksum computations. In some exceptional cases, this option can lead to inaccurate hash computations where one or more checksum validation fails.

  For GRE/IP-in-IP frames, the RSS queue selection is performed using inner headers. The following are possible combinations:

  - GRE/IP-in-IP frame with inner TCP/IP or UDP/IP headers: The RSS is performed using 4-tuple (src IP, dst IP, src port, dst port) hash on the inner IP header and TCP or UDP header.
  - GRE/IP-in-IP frame without inner TCP or UDP header: The RSS is performed using 2-tuple (src IP, dst IP) hash on the inner IP header.
  - Other encapsulated frames (for example, GRE/IP-in-IP frames that cannot be parsed): The RSS is performed using 2-tuple (src IP, dst IP) hash on the outer IP header.

TCP Segmentation Offload

For VXLAN, the TCP Segmentation Offload (TSO) algorithm is performed on the inner TCP segment. The hypervisor provides template TCP/IP headers for the inner TCP segment and template VXLAN/UDP/IP headers for the outer UDP datagram. For every inner TCP segment generated by the VXLAN-aware TSO, the outer VXLAN, UDP, IP, and MAC headers are inserted, and outer IPv4 checksum, outer UDP checksum (not for GRE frames), inner IP checksum (for inner IPv4 datagram only), and inner TCP checksum are computed and inserted. The device updates the IP ID field for every inner TCP segment.

For GRE/IP-in-IP, the Large Segment Offload (LSO) algorithm is performed on the inner TCP segment. The hypervisor provides template TCP/IP headers for the inner TCP segment and template GRE/IP/Ethernet headers for the outer IP datagram. For every inner TCP segment generated by the GRE/IP-in-IP-aware LSO, the outer GRE (not applicable for IP-in-IP frames), IP, and MAC headers are inserted, and outer IPv4 checksum (for outer IPv4 datagrams only), inner IP checksum (for inner IPv4 datagram only), and inner TCP checksum is computed and inserted. The device updates the IP ID field for every inner TCP segment.

Large Receive Offload

Tunneling Offload support for LRO, RSC, TSO, LSO, GSO, and GRO.

SR-IOV Feature for Ethernet Network Adapters

Describes the SR-IOV feature for Ethernet network adapters.

The PCI-SIG defines optional support for Single-Root I/O Virtualization (SR-IOV). SR-IOV allows access to the VM directly to the device using VFs. The network adapter PF is divided into multiple VFs and each VF is presented or bound to the VMs. SR-IOV uses IOMMU functionality to translate PCIe virtual addresses to physical addresses by using a translation table. The number of PFs and VFs is managed through the UEFI HII menu and NVRAM configurations. SR-IOV can be supported in combination with NPAR mode. The SR-IOV feature requires corresponding SR-IOV support in the BIOS (Intel VT-d or AMD IOMMU) and operating system or Hypervisor, as well as the PCIe endpoint device (the network adapter in this case). Broadcom Ethernet network adapters offer the following offload functionality to the VF that is available to the PF:

- TX and RX IP/TCP/UDP checksum offload
- LSO or TCP segmentation offload (TSO, GSO)
- Receive Segmentation Coalescing (RSC) or Large Receive offload (LRO, GRO)
- RSS
- Multiple COS queues
- NVGRE and VXLAN
- RoCEv2

SR-IOV Configuration Support Matrix

For Windows guest operating systems, load the driver specific to the operating system (for example, Win10, Win11, WS22/25).

The following table provides an SR-IOV support matrix.

Table 4: SR-IOV Support Matrix

Host Operating System	Guest Operating System - VF							
	Windows Server 2022	Windows Server 2025*	Windows 10	Windows 11	RH9	RH10	SLES12.5	SLES15.x
Windows Server 2022	Yes	No	Yes	Yes	No	No	No	No
Windows Server 2025	Yes	Yes	Yes	Yes	No	No	No	No
Windows 10	Yes	Yes	Yes	Yes	No	No	No	No
Windows 11	Yes	Yes	Yes	Yes	No	No	No	No
RH9	No	No	No	No	Yes	Yes	Yes	Yes
RH10	No	No	No	No	Yes	Yes	Yes	Yes
SLES12.5	No	No	No	No	Yes	Yes	Yes	Yes
SLES15.x	No	No	No	No	Yes	Yes	Yes	Yes
ESX8	Yes	Yes	No	No	Yes	Yes	Yes	Yes
ESX9	Yes	Yes	No	No	Yes	Yes	Yes	Yes

*Supported Windows guest operating systems for Hyper-V are Windows Server and Azure Stack HCI

Network Partitioning Feature for Ethernet Network Adapters

Describes the Network Partitioning (NPAR) feature for Ethernet network adapters.

Note: Network Partitioning (NPAR) is only supported by BCM5741X and BCM5750X devices.

The Network Partitioning (NPAR) feature allows a single physical network interface port to appear to the system as multiple network device functions. When NPAR mode is enabled, the Broadcom Ethernet network adapter is enumerated as multiple PCIe PFs. Each PF or partition is assigned a separate PCIe function ID on initial power on. Each partition is assigned its own configuration space, BAR address, and MAC address, allowing it to operate independently. Partitions support direct assignment to VMs, VLANs, and so on, just as any other physical interface.

The original PCIe definition allowed for eight PFs per device. For Alternative Routing-ID (ARI) capable systems, the Broadcom Ethernet network adapter supports up to 16 PFs per device.

Security Features for Ethernet Network Adapters

Describes the security features for Ethernet network adapters (BCM5750X and BCM57608 only).

BCM5750X and BCM57608 TruTrust™ technology is capable of secure boot, meaning it only executes boot images authenticated by the secure boot loader (SBL). Secure boot functionality is the cornerstone of a security-enabled system because it is the root of trust from which all subsequent applications are run. The secure boot capability provides the following functionality:

- Secure boot Core Root of Trust – The SBL is based in the device ROM, and outside the scope of modification. The SBL functions as the Core Root of Trust for software, meaning that the system is in a trusted state from reset to when a secure image is authenticated.
- Boot Image Authentication – Only images authenticated by the SBL are executed by the system.
- Boot Image Integrity – The SBL cryptographically validates the integrity of the Secure Boot Image before it is executed to ensure that it has not been tampered with maliciously or errantly.
- Boot Image Confidentiality – The secure processor has the hardware support to execute encrypted images which ensures that device images are never in the clear and protected from reverse engineering or used in device cloning.

Secure devices can be delivered to the customer in a state pending final customization. This customization step is executed by the customer, and once complete, only customer-signed images are executed on the device. Customization provides the following capabilities:

- The customer takes responsibility for the creation and management of keys used in signing their code. This control allows the customer to apply their own security policies in managing their keys and ensures that no code can be signed for their devices by a third party.
- Only code signed by the customer runs on their customized device. This signature ensures that the device code cannot be tampered with in the field and verifies the image authenticity.
- Customer-signed images do not run on other customers' secure devices. This feature prevents piracy of customer images. However, it does not relieve the customer of the responsibility for protecting unsecured binaries from reverse engineering.
- Device or customer-specific encrypted execution images can be generated. This option prevents piracy of customer images and cloning of devices.

RDMA over Converged Ethernet Feature in Ethernet Network Adapters

Describes the RDMA over Converged Ethernet (RoCE) feature and also the requirements for Ethernet network adapters.

Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) is a complete hardware offload feature in the Broadcom Ethernet network adapter which allows RDMA functionality over an Ethernet network. RoCE functionality is available for both user mode and kernel mode applications. RoCE is supported under Linux, Windows, and VMware operating systems.

Note: When RoCE is enabled, the ETS configuration is modified resulting in optimized RDMA performance and less accurate ETS bandwidth allocation.

See the following links for RDMA support for each operating system:

- Windows: Microsoft SMB Direct
- Linux: Installing the Linux Driver
- VMware: VMware Network Requirements for RDMA

Supported Devices for RDMA over Converged Ethernet (RoCE)

The following Ethernet network adapters support RoCE:

- BCM957608
  - BCM957608-P2200GQF00
  - BCM957608-N2200GQP00
  - BCM957608-P1400GDF00
  - BCM957608-N1400GDP00
- BCM95750X
  - BCM957504-P425G
  - BCM957504-N425G
  - BCM957504-N1100G
  - BCM957508-P1200G
  - BCM957508-N1200G
  - BCM957508-P2100G
  - BCM957508-N2100G
- BCM95741X
  - BCM957412A4120AC
  - BCM957412N4120C
  - BCM957412M4123C
  - BCM957416A4160C
  - BCM957416N4160C
  - BCM957416M4163C
  - BCM957414A4142CC
  - BCM957414N4140C
  - BCM957414M4142C
  - BCM957412M4122C
  - BCM957414A4140C
  - BCM957414M4143C

Supported RDMA over Converged Ethernet (RoCE) Protocols

The following RoCE protocols are supported:

- RoCE v2 (default)

Note: RoCE v1 support on Broadcom Ethernet network adapters has been removed.

Supported Operating Systems for RDMA over Converged Ethernet (RoCE)

The following operating systems support RoCE:

Note: The BCM57608 only supports RoCE on the Linux operating system.

- Linux
- VMWare
- Windows

Hardware Requirements for RDMA over Converged Ethernet (RoCE)

Ensure the PCIe slot used for your RNIC supports the Ethernet speed required, otherwise; you will not achieve optimal performance.

Table 5: PCIe Slot Throughput

PCIe Generation	Slot Width (Lanes)	Ethernet Maximum Throughput
Gen 5	16	400 Gb/s
	8	200 Gb/s
Gen4 (Example, AMD Rome/Milan)	16	200 Gb/s
	8	100 Gb/s
Gen 3 (Example, Intel Xeon)	16	100 Gb/s
	8	50 Gb/s

Note: All BCM957608 Ethernet network adapters listed in Supported Devices support PCIe Gen5 x16 speeds. All BCM95750X Ethernet network adapters listed in Supported Devices support PCIe Gen4 x16 speeds and all BCM95741X Ethernet network adapters support PCIe Gen3 x8 speeds.

Ensure that the RNIC receives adequate cooling air from the system fans. Overheating triggers a shutdown, interrupting service. Active cabling, such as AOC, ACC, and optical transceivers, require additional power and produces more heat.

Important Notes

- The PCIe slot width is often written on the system motherboard or PCIe riser next to the slot as xN where N is the width.
- Make note of the maximum network cable speeds. Broadcom Ethernet network adapters typically use SFP and QSFP ports. The selected cable must fit physically and support the maximum speed required. Use the shortest cable that reaches the endpoints for the best speed and lowest bit-error rate.
  - For 25G port speed, use SFP28 DAC or AOC cable
  - For 100G port speed, use QSFP28 DAC or AOC cable
  - For 200G port speed, use QSFP56 DAC or AOC cable
  - For 400G speed, use QSFP112/QSFP56-DD

The following shows a QSP28 AOC cable, 3M in length supporting 100 Gb/s link speed.

Figure 1: Cable Example

Software Requirements for RDMA over Converged Ethernet (RoCE)

The Broadcom RoCE implementation uses the RDMA software stack (librdma, libibverbs) included with all major Linux distributions. Any installation of another vendor's proprietary software stack should be removed before proceeding with these instructions.

Enhanced Network Stack (ENS)

Provides information on Enhanced Network (ENS) LRO feature support in the VMware driver.

Note: Enhanced Network Stack (ENS) is only supported by BCM5741X and BCM5750X devices.

ENS (Enhanced Network Stack) is an enhanced data path in ESXi networking stack mode that provides superior network performance. It is primarily targeted for network virtualization workloads, offering performance benefits leveraging DPDK-like capabilities. The ESXi network stack uses an NSX-T component to enable ENS functionality.

ENS LRO

NSX-T v4.0 and onwards supports LRO (Large Receive Offload) in ENS mode. Broadcom's ESXi 8.0 bnxtnet driver supports this ENS LRO feature. LRO offload is one of the offload features supported by the ESXi driver. When LRO is enabled, the hardware aggregates the packets of the same connection and then notifies the driver. This reduces the number of notifications to the driver, which saves CPU cycles and increases throughput. The following advanced settings must be configured to enable LRO functionality in ENS mode.

A system reboot is not required for the following command to take effect:

```
esxcli system settings advanced set -o /Net/EnsHwLROSupport -i 1
```

To enable LRO with vmknic traffic, use the following command:

```
esxcli system settings advanced set -o /Net/TcpipEnsNetQRSS -i 0
```

To enable LRO with a virtual machine, use the following setting:

```
Set ethernet%d.pnicFeatures=1 or 5 in VM's .vmx file
```

ENS RSS

ESXi 8.0 supports ENSRSS. Since the current ENS stack is not yet populating the information about RSS queues through vsish nodes, the driver logs the enabled number of RSS engines in the vmkernel log. To display more information about each RSS engine, use the following command.

To retrieve the RSS engines configured on ENS uplink:

```
nsxdp-cli ens uplink rss list -n <vmnic>
```

Example output:

```
nsxdp-cli ens uplink rss list -n vmnic7
ID   Type     PriQ NumQ h-Func h-Type keySz  IndSz  SecQs
-----------------------------------------------------------
0    DFTRSS   0    4    2      63     40     128    1 2 3
1    SHARSS   5    4    2      63     40     128    6 7 8
ID: RSS engine Index
Type: DFTRSS = DefaultQ RSS,
    SHARSS = Shared RSS engines(shared by multiple clients),
    OFLRSS = Offloaded RSS/Dedicated RSS(dedicated to single VNIC)-Not supported
```

Set the kernel parameters as follows to enable ENS RSS and/or LRO features and use the latest ESXi 8.0 GA builds.

For vmknic RSS + LRO enable:

```
--------------------------
esxcli system settings advanced set -o /Net/TcpipEnsMultipleRxContexts -i 1
esxcli system settings advanced set -o /Net/EnsHwLROSupport -i 1
esxcli system settings advanced set -o /Net/TcpipEnsDedicatedNetQRSS -i 1
```

For VM RSS + LRO enable:

```
----------------------
esxcli system settings advanced set -o /Net/EnsHwLROSupport -i 1
Modify ethernet%d.rssOffload to 1(Dedicated RSS) and ethernet%d.pnicFeatures=5 in VM's .vmx file
```

For vmknic RSS only:

No extra specific parameters need to be set, as it is enabled by default.

For VM RSS only:

No extra specific parameters need to be set, as by default ethernet%d.pnicFeatures is 4(RSS only) in .vmx VM's file

For vmknic LRO only:

```
--------------------
esxcli system settings advanced set -o /Net/TcpipEnsNetQRSS -i 0
esxcli system settings advanced set -o /Net/EnsHwLROSupport -i 1
```

For VM LRO only:

```
----------------
esxcli system settings advanced set -o /Net/EnsHwLROSupport -i 1
Set ethernet%d.pnicFeatures=1 in VM's .vmx file.
```

Ensure that the previous options are set before creating dvSwitch.

Note: Anytime a vmknic or vnic port in a DVS requests shared RSS, the whole DVS is considered as using shared RSS. Therefore, LRO would not take effect. If there are multiple vmknic and/or vnic ports in the DVS and there is a need to enable LRO (either just LRO or LRO+RSS), ensure all ports have settings for either LRO only or LRO+(dedicated)RSS.

Multi RSS

Multi RSS in the ESXi driver is controlled by the `enable_multi_rss` module parameter. Multi-RSS is enabled by default and when enabled, the driver tries to enable 4 RSS engines.

In ENS mode, if possible, the RSS engine is created with 8 RSS queues by default. If not enough resources are available, the driver enables RSS with either a smaller number of engines and/or a smaller number of RSS queues in both ENS and Non-ENS mode.

Limitations:

1. The ESXi driver disables Multi-RSS capability if not enough resources are available. Some of the possible scenarios where this could happen are when NPAR or NPAR + SRIOV is enabled and/or on servers with fewer system resources (fewer CPU cores). In this case, the driver logs the appropriate message in the vmkernel log.
2. The ESXi driver enables Multi-RSS capability with a smaller number of RSS engines and a smaller number of RSS queues on BCM9574XX/BCM9575XX NPAR, NPAR + SRIOV, setups as the hardware resources are limited in this configuration. Driver logs the appropriate message in the vmkernel log.

XDP and AF_XDP

Provides information on XDP and AF_XDP support for the Linux driver.

XDP (eXpress Data Path) and AF_XDP are supported within the Linux driver for Broadcom Ethernet network adapters.

XDP Support and XDP-Bench

XDP provides a high-performance, programmable network data path in the Linux kernel. XDP-bench is a reference tool that can be used to determine data performance. It can be downloaded here.

XDP-Bench Drop Command

In this mode, XDP-bench installs an XDP program on an interface that drops all packets. There are options to control what to do with the packet before dropping it (touch the packet data or not), as well as which statistics to gather. This is a basic benchmark for the baseline (best-case) performance of XDP on an interface. Example:

```
xdp-bench drop [options] <ifname>
```

<ifname> is the name of the interface the XDP program is installed on.

AF_XDP

AF_XDP is an address family that is optimized for high-performance packet processing. For a complete overview of AF_XDP, refer to the kernel documentation located here.

AF_XDP Sample Application

An xdpsock benchmarking/test application is included that demonstrates how to use AF_XDP sockets with private UMEMs. For example, if it is required for UDP traffic from port 4242 to end on queue 16 and enable AF_XDP, use ethtool for this as follows:

```
ethtool -N p3p2 rx-flow-hash udp4 fn
ethtool -N p3p2 flow-type udp4 src-port 4242 dst-port 4242 \
action 16
```

Running the rxdrop benchmark in XDP_DRV mode can then be accomplished using the following command:

```
samples/bpf/xdpsock -i p3p2 -q 16 -r -N
```

Supported Feature Combinations for Ethernet Network Adapters

Describes the supported feature combinations for Ethernet network adapters.

The following table shows the supported feature combinations of NPAR, SR-IOV, and RoCE.

Table 6: Supported Combinations

SW Feature	RoCE	DPDK
NPAR	Up to 8 PFs or 16 PFs	Up to 8 PFs or 16 PFs
SR-IOV	Up to 128 VFs (total per chip)	Up to 128 VFs (total per chip)
RoCE on PFs	Up to 4 PFs for the BCM5741X devices and 16 PFs for BCM5750X devices	–
RoCE on VFs	BCM5750X supports RoCE SR-IOV up to 128 VFs since the 219.x release. This is for single-host adapters only. Multiroot adapters are limited to 48 VFs. The 218.x release does not support RoCE SR-IOV for BCM5750X. BCM5741X does not support RoCE on VFs.	–
Host Operating System	Linux, Windows, ESXi (no vRDMA support)	Linux
Guest Operating System	Linux and Windows	DPDK (Linux)
DCB	Up to eight COS per port	–

Note: Starting with the 228 software release, the Linux driver has increased the default ring allocation for network interfaces. The total number of combined rings across all network interfaces on a given Ethernet adapter is now limited to the number of near-NUMA CPUs, with a maximum cap of 64 rings. This adjustment is intended to enhance the out-of-the-box performance of the Ethernet adapter. When specific NVM configuration options, such as RoCE and/or NPAR, are enabled on the adapter, this maximum value is restricted to 16 rings per PF to prevent potential resource exhaustion. However, if an insufficient resource issue is encountered when SR-IOV is enabled along with NPAR or RoCE, it is recommended to reconfigure the PF with only 8 rings (using ethtool -L <interface> combined 8 tx 0 rx 0.) and then retry enabling SR-IOV.

Note: Certain 4-port BCM95750X adapters support up to 32 VF and 4 NPAR per port. When both NPAR and SR-IOV are enabled in the ESXi operating system, users may not configure more than 8 VFs on each partition.

Unsupported Features for Ethernet Network Adapters

Describes the unsupported feature combinations for Ethernet network adapters.

- RoCE SR-IOV + NPAR is not supported.
- The BCM5741X does not support RoCE SR-IOV.
- The BCM5750X does not support RoCE SR-IOV in 218.x release. Releases starting with 219.x support RoCE SR-IOV for the BCM5750X.

Installing Broadcom Ethernet Network Adapter Hardware

Provides instructions for installing the hardware components of Broadcom Ethernet network adapters.

All components are bundled in the zip file available from the individual Ethernet Network Adapter device pages on Broadcom.com. The documentation link contains the driver zip file.

This section provides the following hardware installation information:

- Broadcom Ethernet Network Adapter Safety Precautions
- Hardware Requirements for Broadcom Ethernet Network Adapters
- Installing the Ethernet Network Adapter
- Cable/Interconnect Compatibility list for Broadcom Ethernet Network Adapters
- UEFI HII Menu for Ethernet Network Adapters

Broadcom Ethernet Network Adapter Safety Precautions

Provides safety precautions for the Ethernet network adapter.

CAUTION: Server class system power supplies with higher current can be hazardous when normal operating procedures are not used. Before removing the system cover, observe the following precautions to protect yourself and prevent damage to the system components:

- Remove any metallic objects or jewelry from your hands and wrists.
- Make sure to use only insulated or non-conducting tools.
- Verify that the system is powered OFF and unplugged before you touch internal components.
- Install or remove adapters in a static-free environment. The use of a properly grounded wrist strap or other personal antistatic devices and an antistatic mat is strongly recommended.

Hardware Requirements for Broadcom Ethernet Network Adapters

Provides hardware requirements for Broadcom Ethernet network adapters.

Ensure your system meets the following hardware requirements for Ethernet network adapters:

- One open PCIe Gen3, Gen4, or Gen5 slot in x8 or x16 mode.
  - BCM95741XA41XX – PCIe Gen3 slot in x8 mode.
  - BCM95741XM41XX – OCP 2.0 or rNDC slots.
  - BCM957XXXNXXXX – OCP 3.0 slot
  - BCM95750X-PXXXX – PCIe Gen3 or Gen4 slot in x16 mode.
  - BCM95750X-MXXXX – OCP 2.0 slot in x16 mode
  - BCM957608-PXXXX – PCIe Gen5 slot in x16 mode.
  - BCM957608-NXXXX – OCP 3.0 slot in x16 mode
- 16 GB memory or more (use 32 GB or more for virtualization applications and nominal network throughput performance).

Replacing the Ethernet Network Adapter Mounting Brackets

Provides information on replacing a standard-profile bracket with a low-profile bracket.

Use the following steps to replace the standard-profile mounting bracket that ships on the adapter with a low-profile bracket.

1. Using a #0 Phillips screwdriver that is ESD safe, remove the two Phillips screws that connect the full-profile bracket to the adapter. Avoid touching any board components with the screwdriver or the bracket.
2. Remove the full-profile bracket. Do not damage the adapter.
3. Place the adapter on top of the low-profile bracket and position the bracket so that the screw holes in the tabs align with the openings on the adapter.
4. Using a Phillips torque screwdriver that is ESD safe, set to a maximum torque of 2.0 kgf.com. Replace the two Phillips screws removed in step 1.

Caution: Exceeding this torque specification can damage the adapter, connectors, or screws, and can void the warranty on the adapter.

Caution: Damage caused to the adapter as a result of changing the bracket can void the warranty on the board. Adapters returned without a bracket mounted on the board will be sent back without return merchandise authorization (RMA) processing.

Caution: Ensure the LEDs/Light pipes are aligned with the corresponding holes on the bracket.

Installing the Ethernet Network Adapter

Provides instructions for installing Broadcom Ethernet network adapters.

Use the following steps to install the Broadcom Ethernet network adapter (add-in NIC) into most servers. See the manuals that are supplied with the server for details about performing these tasks on a particular server.

1. Review the Safety Precautions and Preinstallation Checklist before installing the adapter. Ensure that the system power is completely OFF and unplugged from the power outlet, and that proper electrical grounding procedures have been followed.
2. Open the system case and select any empty PCIe Gen3 x8 slot, Gen4 x16 slot, or Gen5 x16 slot.
3. Remove the blank cover plate from the slot.
4. Align the adapter connector edge with the connector slot in the system.
5. Secure the adapter with the adapter clip or screw.
6. Close the system case and disconnect any personal antistatic devices.

Note: When inserting or removing an adapter, do not use the network connectors or heatsink of the adapter for leverage.

Cable/Interconnect Compatibility List for Broadcom Ethernet Network Adapters

Provides a list of supported interconnects for Broadcom Ethernet network adapters.

The interconnect compatibility lists are divided into the following sections:

- Cable/Interconnect Compatibility for BCM9575XX, BCM9574XX, and Earlier Adapters
- Cable/Interconnect Compatibility for BCM957608 (Thor 2) Adapters

Cable/Interconnect Compatibility for BCM9575XX, BCM9574XX, and Earlier Adapters

Provides a list of supported interconnects for the BCM9575XX, BCM9574XX, and earlier adapters.

Broadcom executes basic interoperability testing with a subset of cables and transceivers from the marketplace. The interoperability testing does not include BER tests, power, or temperature measurements. Broadcom validates interoperability with many different AOCs, optical modules, and DAC cables. Broadcom Ethernet adapters support any IEEE-compliant AOC, optical module, or DAC cable in the market, even if a particular AOC, optical module, or DAC cable might not have been part of a qualification cycle within Broadcom's interoperability lab. Moreover, Broadcom ensures continued compliance with these industry standards by constantly testing with a subset of industry-compliant AOCs, optical modules, and DAC cables.

The firmware code conforms to the following SNIA industry standards:

- SFF-8024 (SFF Module Management Reference Code Tables)
- SFF-8472 (Management Interface for SFP+)
- SFF-8636 (Management Interface for 4-lane Modules and Cables)
- SFF-8665 (QSFP+ 28 Gb/s 4× Pluggable Transceiver Solution (QSFP28))

The cables shown in the following table are tested for compatibility with Broadcom's generic release or later.

Note: Broadcom cannot test all cables that are available at the time of this publication. Cables adhering to the previous SNIA specifications should operate normally; however, Broadcom cannot guarantee compatibility.

Table 7: Tested Cables

Manufacturer	Cable/Transceiver Part no	Speed	Connector	Type	Length
3M	9QM6-517-23-2.75	100G	QSFP28	DAC	N/A
3M	9QM6-517-23-2.50	100G	QSFP28	DAC	N/A
Accelink	RTXM330-005-C20	25G	SFP28	AOC	20 m
Accelink	RTXM330-C05	25G	SFP28	AOC	5 m
Accelink	RTXM330-C07	25G	SFP28	AOC	7 m
Accelink	RTXM330-C10	25G	SFP28	AOC	10 m
Accelink	RTXM330-C20	25G	SFP28	AOC	20 m
Accelink	RTXM420-550	100G	QSFP28	Transceiver	N/A
Accelink	RTXM290-806-C14	100G	–	Transceiver	N/A
Amphenol	038-003-728-01	10G	SFP+	DAC	1 m
Amphenol	038-004-142-01	10G	SFP+	DAC	2 m
Amphenol	038-003-729-01	10G	SFP+	DAC	3 m
Amphenol	038-003-730-01	10G	SFP+	DAC	5 m
Amphenol	038-004-931-01	25G	SFP28	DAC	1 m
Amphenol	038-004-933-01	25G	SFP28	DAC	2 m
Amphenol	038-004-935-01	25G	SFP28	DAC	3 m
Amphenol	FOQQD33P00003	100G	QSFP28	AOC	3 m
Amphenol	038-004-926-00	100G	QSFP28	DAC	1 m

Manufacturer	Cable/Transceiver Part no	Speed	Connector	Type	Length
Amphenol	038-004-928-00	100G	QSFP28	DAC	2 m
Amphenol	038-004-930-00	100G	QSFP28	DAC	3 m
Amphenol	NDAAXJ0003	200G	QSFP56	DAC	3 m
Amphenol	NDARXJ0003	200G	QSFP56	DAC	3 m
Amphenol	NDARHG-F306	200G	QSFP56	DAC	3 m
Arista Networks	SFP-10G-SR	10G	SFP+	Transceiver	N/A
Arista Networks	AOC-S-S-10G-5M	10G	SFP+	AOC	5 m
Arista Networks	CAB-SFP-SFP-5M	10G	SFP+	DAC	5 m
Arista Networks	SFP-25G-SR	25G	SFP28	Transceiver	N/A
Arista Networks	SFP-25G-MR-SR	10/25G	SFP28	Transceiver	N/A
Arista Networks	CAB-S-S-25G-5M	10/25G	SFP28	DAC	5 m
Arista Networks	CAB-Q-4S-100G-5M	10/25G	SFP28	DAC	5 m
Arista Networks	QSFP-100G-DR	100G	QSFP28	Transceiver	N/A
Arista Networks	QSFP-100G-SR4	100G	QSFP28	Transceiver	N/A
Arista Networks	QSFP-100G-SRBD	100G	QSFP28	Transceiver	N/A
Arista Networks	AOC-Q-Q-100G-5M	100G	QSFP28	AOC	5 m
Arista Networks	CAB-Q-Q-100G-5M	100G	QSFP28	DAC	5 m
Arista Networks	C-Q200-2Q100-3M	50/100G	QSFP28	DAC	3 m
Arista Networks	CAB-D-2Q-400G-3M	100/200G	2XQSFP-56	DAC	3 m
Avago	AFBR-710DMZ	10G	SFP+	Transceiver	N/A
Avago	AFBR-710SMZ	10G	SFP+	Transceiver	N/A
Avago	AFBR-735SMZ	10/25G	SFP28	Transceiver	N/A
Avago	AFBR-8CER02Z	10/25G	SFP28	AOC	2 m
Avago	AFBR-89CDHZ	100G	QSFP28	Transceiver	N/A
Avago	AFBR-89BDDZ	100G	QSFP28	Transceiver	N/A
AXIOM	QSFP-100G-C010-AX	100G	QSFP28	AOC	10M
AXIOM	QSFP-100G-AOC7M-AX	100G	QSFP28	AOC	7M
AXIOM	470-ABOX-AX	100G	QSFP28	DAC	3M
Broadex	DHZZJJ-KCCCC-010	200G	QSFP56	DAC	1 m
Broadex	DHZZJJ-KCCCC-020	200G	QSFP56	DAC	2 m
Cisco	SFP-10G-AOC1M	10G	SFP+	AOC	1 m
Cisco	SFP-10G-AOC7M	10G	SFP+	AOC	7 m
Cisco	SFP-H10GB-CU7M	10G	SFP+	DAC	7 m
Cisco	SFP-25G-AOC7M	25G	SFP28	AOC	7 m
Cisco	SFP-H25G-CU5M	25G	SFP28	DAC	5 m
Cisco	SFP-25G-AOC5M	25G	SFP28	AOC	5 m
Dell	C5RNH	10G	SFP+	Transceiver	N/A

Manufacturer	Cable/Transceiver Part no	Speed	Connector	Type	Length
Dell	K0T7R	10G	SFP+	AOC	15 m
Dell	WTRD1	10G	SFP+	Transceiver	N/A
Dell	YXT4J/APSP 831B53IDL10	10G	SFP+	Transceiver	N/A
Dell	4G3TH/APSP CxxHM3IDL70	10G	SFP+	Transceiver	N/A
Dell	J9CJV/APSP DxxHM3IDL80	10G	SFP+	Transceiver	N/A
Dell	2NPMJ/APSP 31B33CDL10	10G	SFP+	Transceiver	N/A
Dell	M14MK	10/25G	SFP28	Transceiver	N/A
Dell	P7D7R	25G	SFP28	Transceiver	N/A
Dell	3YWG7	25G	SFP28	AOC	7 m
Dell	9X8JP	25G	SFP28	DAC	5 m
Dell	0YR96	25G	SFP28	Transceiver	N/A
Dell	RCVP5	25G	SFP28	AOC	15 m
Dell	1HW01/APS8 B23B53CDL10	25G	SFP28	Transceiver	N/A
Dell	DT7JH/APS8 DxxB53IDL10	25G	SFP28	Transceiver	N/A
Dell	C5M6G/APS8 C27B53CDL10	25G	SFP28	Transceiver	N/A
Dell	VXFJY	25G	SFP28	DAC	3 m
Dell	4WGYD	100G	QSFP28	Transceiver	N/A
Dell	407-BBZL	100G to 25G	QSFP28 to SFP28	Transceiver Adapter	N/A
Dell	YTCF3	100G	QSFP56DD to 4xQSFP56	DAC	1 m
Dell	TVFDN	100G	QSFP56DD to 4xQSFP56	DAC	2 m
Dell	GP3HN	200G	QSFP56DD to 2xQSFP56	DAC	1 m
Dell	83MHF	200G	QSFP56-DD to 2xQSFP56	DAC	1 m
Dell	3HDHY	200G	QSFP56DD to 2xQSFP56	DAC	2 m
Dell	JJ3PN	200G	QSFP56-DD to 2xQSFP56	DAC	2 m
Dell	G001T	200G	QSFP56DD to 2xQSFP56	DAC	3 m
Dell	R1K01	200G	QSFP56-DD to 2xQSFP56	DAC	3 m
Finisar	FTLX8574D3BCV	10G	SFP+	Transceiver	N/A
Finisar	FTLX8571D3BNL E5 REV.B1	10G	SFP+	Transceiver	N/A

Manufacturer	Cable/Transceiver Part no	Speed	Connector	Type	Length
Finisar	FCCG125SD1C10BTE	25G	SFP28	AOC	10 m
Finisar	FCBG125SD1C05BTE	25G	SFP28	AOC	5 m
Finisar	FCCG125SD1C05	25G	SFP28	AOC	5 m
Finisar	FCCG125SD1C20BTE	25G	SFP28	AOC	20 m
Finisar	FTLF8536P4B NL-E5 REV.B1	25G	SFP28	Transceiver	N/A
Finisar	FCBN425QE1C03	100G	QSFP28	AOC	3 m
Finisar	FTLC9551REPM	100G	QSFP28	Transceiver	N/A
Finisar	FTLC9558REPM	100G	QSFP28	Transceiver	N/A
Finisar	FTLC9555REPM3-E5	100G	QSFP28	Transceiver	N/A
Finisar	FTLC1154RDPL	100G	QSFP28	Transceiver	N/A
Gigalight	GQS-MPO201-SR4C	200G	QSFP56	Transceiver	N/A
H3C	SFP-25G-D-A OC-10M-DG	25G	SFP28	AOC	10 m
H3C	SFP-25G-D-A OC-20M-DG	25G	SFP28	AOC	20 m
H3C	SFP-25G-D AOC-5M-DG	25G	SFP28	AOC	5 m
H3C	SFP-25G-D AOC-7M-DG	25G	SFP28	AOC	7 m
H3C	QSFP-100G-S R4-MM850-A	100G	QSFP28	Transceiver	N/A
H3C	QSFP-100G-CW DM4-SM1300-A	100G	QSFP28	Transceiver	N/A
HiSense	DMQ8611A-DC03	200G	QSFP56	AOC	3 m
HiSense	LMQ8611A-PC+	200G	QSFP56	Transceiver	N/A
HPE	AP818A	10G	SFP+	AOC	1 m
HPE	844477-B21	25G	SFP28	DAC	3 m
HPE	845406-B21	100G	QSFP28	Cable	3 m
HPE	845970-B21	100G to 25G	QSFP28 to SFP28	Transceiver Adapter	N/A
Huawei	QSFP-100G-SR4	100G	QSFP28	Transceiver	N/A
Innolight	TF-PY005-NTA	25G	SFP28	AOC	5 m
Innolight	TF-PY020-NTA	25G	SFP28	AOC	20 m
Innolight	TR-FC85S-NTC	100G	QSFP28	Transceiver	N/A
Innolight	TR-FC13T-NTC	100G	QSFP28	Transceiver	N/A
Innolight	TR-FC13R-N00	100G	QSFP28	Transceiver	N/A
Luxshare	SFP-25G-DA C-5.0M-B-TX	25G	SFP28	DAC	5 m
Luxshare	SFP-25G-ACC-7.0M	25G	SFP28	AOC	7 m
Luxshare	PA02SMA06-SD-R	25G	SFP28	Transceiver	100 m
Luxshare	PA0SS2302-SD-R	25G	SFP28	AOC	3 m
Luxshare	PAO1QMA01-SD-R	100G	QSFP28	Transceiver	100 m
Luxshare	PA0QQ1312-SD-R	100G	QSFP28	AOC	3 m

Manufacturer	Cable/Transceiver Part no	Speed	Connector	Type	Length
Mellanox	MFA2P10-A005	10/25G	SFP28	AOC	5 m
Mellanox	7G17A03537/ MMA2P00-AS	25G	SFP28	Transceiver	N/A
Mellanox	MMA1B00-C100D	100G	QSFP28	Transceiver	N/A
Mellanox	MFA1A00-C005	100G	QSFP28	AOC	5 m
Mellanox	MFA7A20-C003	100G	QSFP28	AOC	3 m
Mellanox	4Z57A14197/ M F51550-H005E	200G	QSFP56	AOC	5 m
Mellanox	MFS1S00-V003E	200G	QSFP56	AOC	3 m
Mellanox	MMA1T00-VS	200G	QSFP56	Transceiver	N/A
Mellanox	MFS1S00-V005E	200G	QSFP56	AOC	5 m
Mellanox	845969-001	100G to 25G	QSFP28 to SFP28	Transceiver Adapter	N/A
Molex	1002976020	100G	QSFP28	DAC	2 m
Molex	1002976007	100G	QSFP28	DAC	0.75 m
Molex	1002976010	100G	QSFP28	DAC	1 m
Molex	1002976030	100G	QSFP28	DAC	3 m
Molex	1002976035	100G	QSFP28	DAC	3.5 m
Optomind	C4R448GA005AZZ	200G	QSFP56	AOC	5 m

For further information, contact your local Broadcom field or application engineer.

Cable/Interconnect Compatibility for BCM957608 (Thor 2) Adapters

Provides a list of supported interconnects for BCM957608 (Thor 2) adapters.

The Broadcom Interconnect Compatibility List (ICL) provides a list of supported interconnects, including DAC and Optical solutions for Broadcom BCM9576XX (Thor 2) Ethernet network adapters.

- Broadcom executes basic interoperability testing with a subset of interconnects from the marketplace in DAC, AOC, and Optical Transceivers. The compatibility focuses on functionality and does not include performance or thermal testing.
- Broadcom Ethernet adapters are generally expected to work with any IEEE-compliant DAC, AOC, or Optical Transceivers, even if a particular interconnect might not have been part of a qualification cycle within Broadcom's interoperability lab.
- Broadcom ensures continued compliance with these industry standards by constantly testing with a subset of industry-compliant interconnects.
- Broadcom is actively working and qualifying DAC cables that exceed the IEEE spec (>2M for 112G rates); these cables can be found on the ICL

The following table is used to determine the best cable configuration using the following steps:

1. Determine Switch SerDes speed: 56G or 112G (if slower use 56G).
2. Determine Switch Connector Type (OSFP, QSFP-DD, and so forth).
3. Determine cable type: DAC for shorter distances and lower power. Optical for longer distances.
4. Determine required Port Configuration (1x400, 2x200, and so forth).

Using this information, find the appropriate cable configurations in the following table. Download the BRCM9576XX Cable ICL and using the Broadcom Cable Configuration Guide, filter for Broadcom-qualified cables.

Table 8: Broadcom BCM957608 (Thor 2) Cable Configurations

Switch	Type	Cable Configuration	Ethernet Port Configuration	Switch-Side Connector (SS)	Switch:NIC Split	NIC-Side Connector (NS)
Tomahawk 4 or other 56G Switch	DAC (1-5m)	D1	2x200G	QSFP56-DD	1:2	QSFP56
		D2	1x400G	QSFP56-DD	1:2	QSFP56
		D3	1x400G	QSFP56-DD	1:1	QSFP56-DD
		D4	1x200G	QSFP56	1:1	QSFP56
		D5	2x200G	QSFP56-DD	2:2	QSFP56-DD
	Optics (non-LPO) (1m-kms)	01 (Gearbox)	1x400G	QSFP56-DD	1:1	QSFP112
		02	2x200G	QSFP56-DD	1:2	QSFP56
		03	1x400G	QSFP56-DD	1:1	QSFP56-DD
Tomahawk 5 or 112G Switch	DAC (1-5m)	D21	2x400G	OSFP112	1:2	QSFP112
		D22	1x400G	QSFP112	1:1	QSFP112
	Optics (non-LPO) (1m-kms)	021	2x400G	OSFP112	1:2	QSFP112
		022	1x400G	QSFP112	1:1	QSFP112
	Optics (LPO) (1m-kms)	L21	2x400G	OSFP112	1:2	QSFP112

For more details on cable types, connectors, and configurations download the Broadcom Cable Configuration Guide.

- BCM957608 Interconnect Compatibility List (ICL)
- BCM957608 Cable Solutions Guide (957608-AN1XX)

UEFI HII Menu for Ethernet Network Adapters

Provides configuration instructions for Ethernet network adapters using the Human Interface Infrastructure (HII) menu at boot time.

Note: UEFI mode is supported on x86 systems and boards that support ARM architecture.

This menu system allows the configuration of all persistent settings such as boot protocol (PXE), virtualization modes (SR-IOV, NPAR), and so on. To enter the HII configuration menu, follow boot-time prompts to BIOS, and then device configuration. The menu layout of the adapters might differ. Some settings might be unavailable for some adapter types.

Note: For BCM957412-N410TGP0 and BCM957412-P410TGP0 devices with dual-chips, all configuration settings must be configured on both chips. Chip1 = Port 1 and Port 2. Chip2 = Port 3 and Port 4.

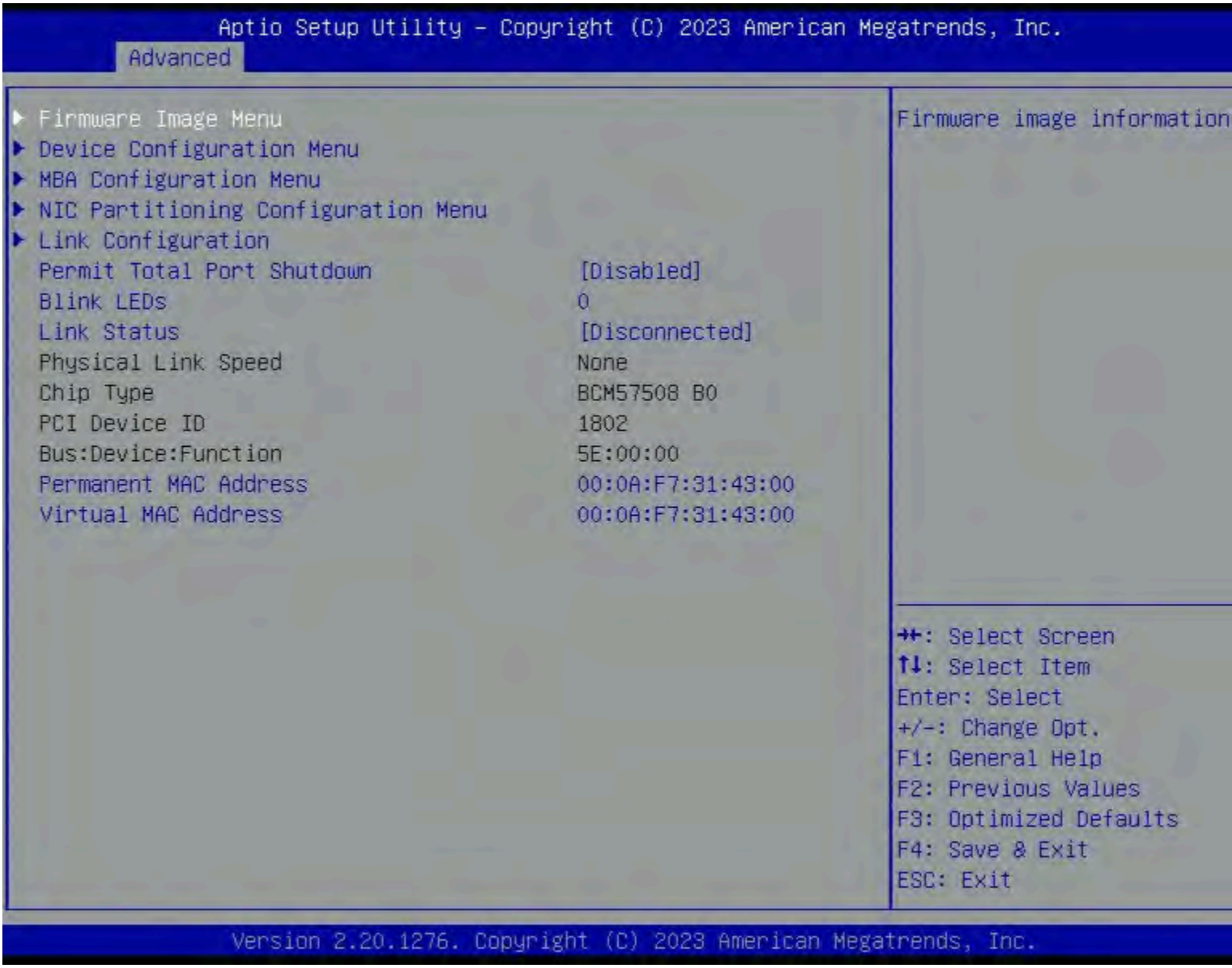
The UEFI HII configuration consists of the following menus:

- Main Configuration Page
- Firmware Image Menu
- Device Configuration Menu
- MBA Configuration Menu NIC Configuration
- NIC Partitioning Configuration Menu
- Partition n Configuration Menu

- Link Configuration Menu

Main Configuration Page

Figure 2: Main Configuration Page



This page displays the following information:

- **Firmware Image Menu** – This menu presents the various component versions present in the current firmware package.
- **Device Configuration Menu** – This menu presents adapter-specific parameters for configuration.
- **MBA Configuration Menu** – This menu presents PXE boot-related parameters for configuration.
- **NIC Partitioning Configuration Menu**– This menu presents NIC partition-related parameters for configuration.
- **Link Configuration** – This menu presents link-related parameters for configuration.

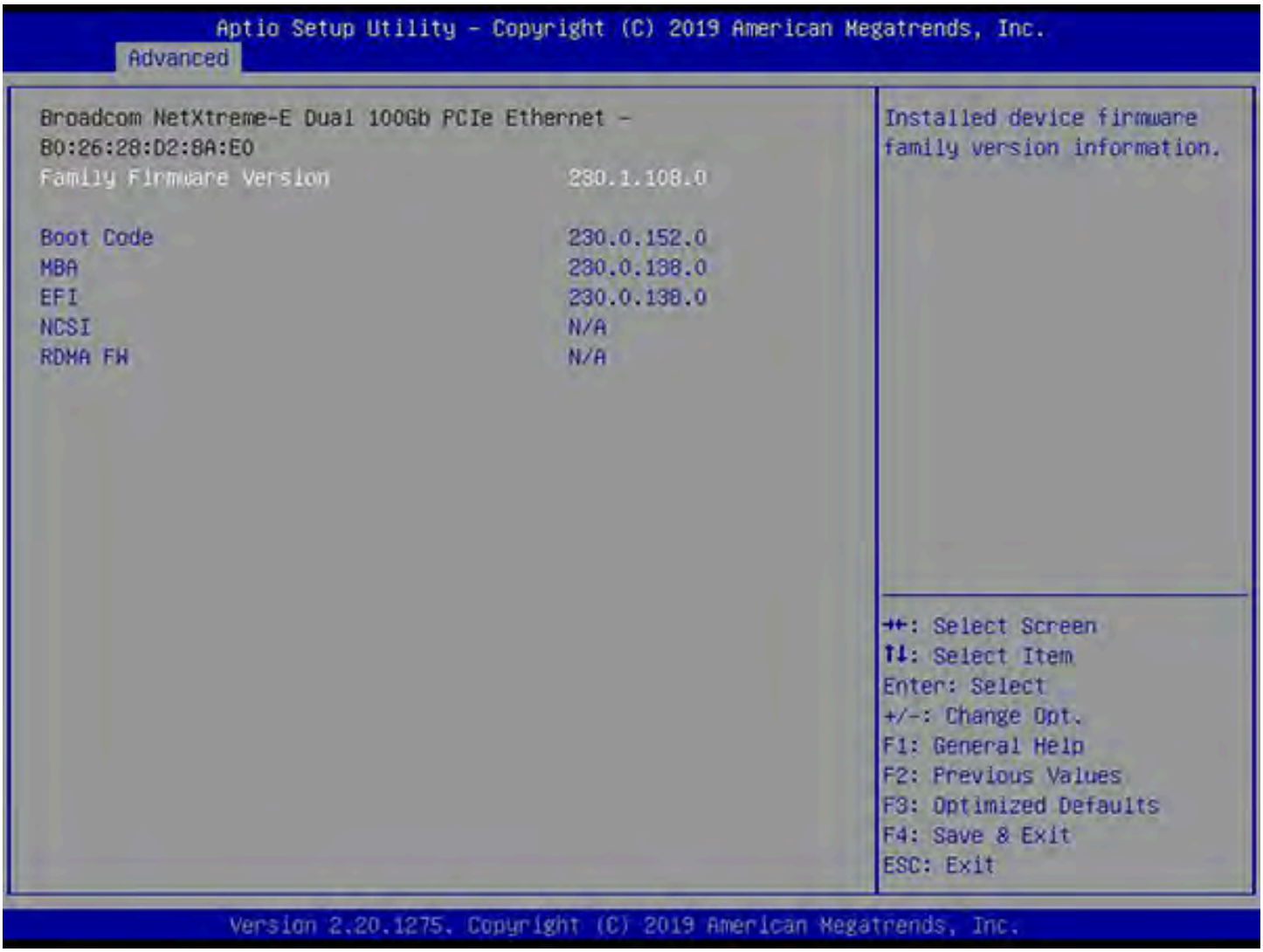
- **Permit Total Port Shutdown** – Specifies whether or not to allow the port to be completely disabled when a Port Down command is received from the Host OS or driver. This option is only supported when Virtualization Mode is None or SR-IOV, and on Linux operating systems only.

  Note: Port shutdown halts all operations configured on the port including Wake On LAN and shared LOM.
- **Blink LEDs** – Configures the duration for which the LEDs on the physical network port should blink to assist with port identification. This is a numeric setting. The value must be specified in the range of 0 to 15 seconds.
- **Link Status** – Displays the physical link status of the network port as reported by the controller. This is a read-only field.
  - **Connected** – Link is up
  - **Disconnected** – Link is down
- **Physical Link Speed** – Displays the current link speed of the network port as reported by the controller. This is a read-only field. Speed is reported in Mb/s/Gb/s.
- **Chip Type** – Displays the Broadcom-specific identifier which denotes the adapter family to which the chip belongs and the revision. This is a read-only field.
- **PCI Device ID** – Displays the 16-bit PCI Device ID reported by the controller. This is a vendor-defined ID that varies across non-NPAR, NPAR, and RDMA mode. Refer to MF mode and Support RDMA sections for more information on these modes. This is a read-only field.
- **Bus Device Function** – Displays the BIOS assigned PCI Bus:Device:Function identifier of the card. This is a read-only field.
- **Permanent MAC Address** – Displays the Permanent MAC address assigned during manufacturing. This is a read-only field.
- **Virtual MAC Address** – Displays the Virtual MAC address assigned to the device. This is a read-only field from the HII menu. The value for this parameter can be configured using remote utilities.

Firmware Image Menu

This menu presents the various component versions present in the current firmware build. Depending on the adapter type, some components may not be available. All fields in this menu are read-only.

Figure 3: Firmware Image Menu



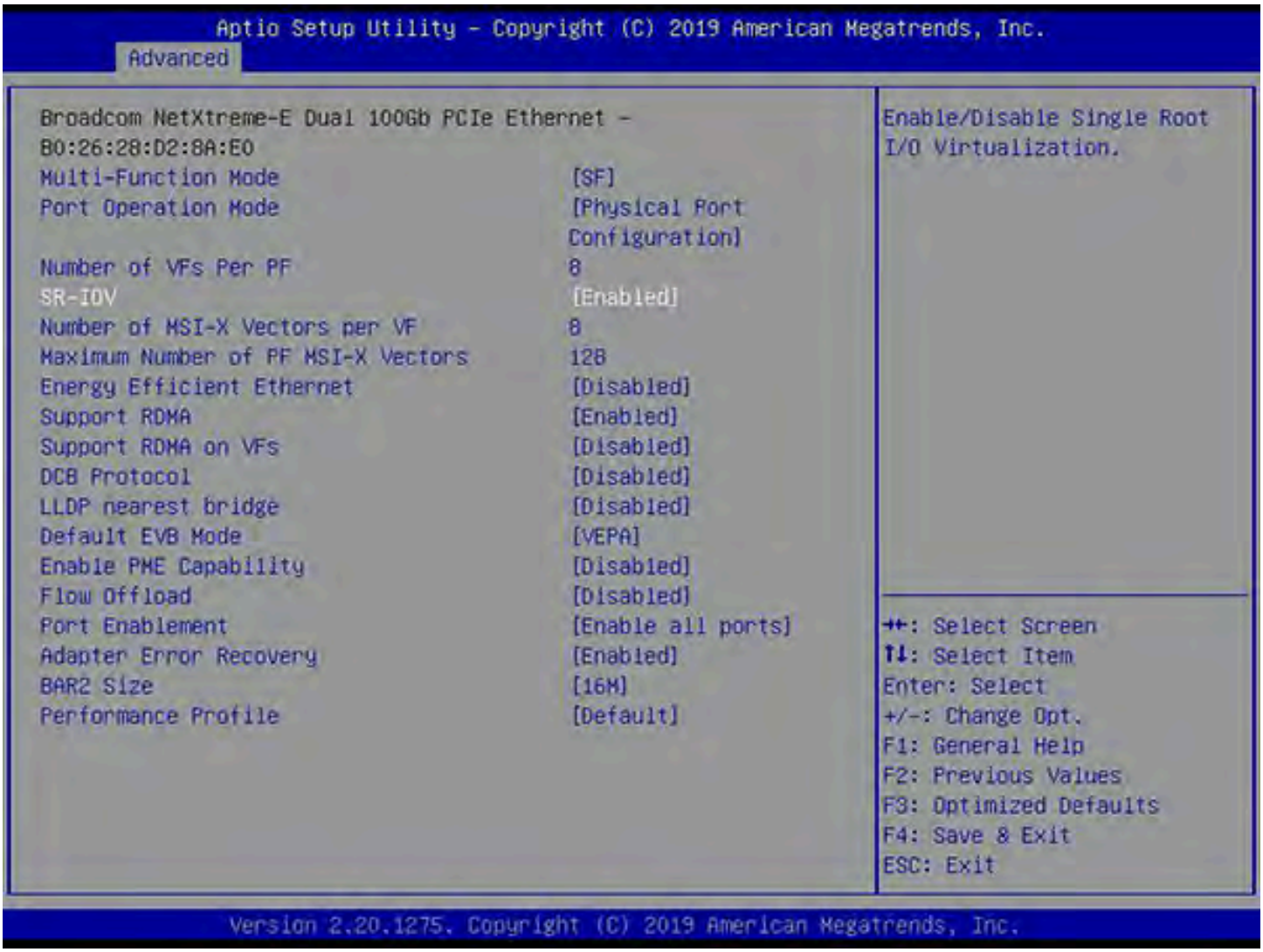
This page displays the following information:

- **Family Firmware Version** – Displays the family firmware version. This field may be displayed as Firmware Bundle on some adapters.
- **Boot Code** – Displays the firmware boot code version.
- **MBA** – Displays the legacy pre-boot driver version.
- **EFI** – Displays the UEFI pre-boot driver version.
- **NC-SI** – Displays the NC-SI firmware version.
- **RDMA FW** –Displays the RoCE firmware version.

Device Configuration Menu

This menu presents adapter-specific parameters for configuration. Depending on the adapter type, some settings may not be available.

Figure 4: Device Configuration Menu



This page allows the user to configure the following items:

- **Multi-Function Mode** – Configures the type of virtualization to be used by the controller on all ports. This is available only when NPAR is supported on the adapter.
  - **Single Function Mode (SF)** – In this mode, a single PCIe PF is assigned to each network port.
  - **Network Partitioning Mode (NPAR)** – Allows a single physical network port to appear to the system as multiple network device functions. Each PF or partition is assigned a separate PCIe function ID on initial power on, and a menu is made visible in setup to configure each partition. The 16 configurable partitions are distributed equally across the network ports on an ARI-capable system.
- **Port Operation Mode** (**BCM95750X/BCM957608 Only)** – This setting allows the user to configure the operational mode of the ports on the device.

Note: Before changing the **Port Operation Mode**, it is recommended that the NVM cfg file be reset to the factory default settings. Resetting the NVM cfg file can be done using the NICCLI command (see NICCLI Configuration Utility Usage and Commands).

- **Physical Port Configuration** - This option sets the device's port operation mode to the original physical board design.
- Port Breakout Configuration - Configuring Port Breakout on BCM95750X/BCM957608 Ethernet Adapters
- **Disable Port 2** - This option disables the functionality of the physical port(s) Px on the device.

  Note: To disable port 2 on BCM95750X devices, see **Port Enablement**. The **Disable Port 2** option is not available in BCM95750X devices.
- **Aggregate 2 ports to 1 and 4 ports to 1** - This option combines X physical ports into Y functional port(s).

Note: The BCM957608 supports aggregation on two 200G ports to achieve 400G. No other aggregation types are supported for the BCM957608. For example, two 100G ports to achieve 200G is not supported.

- **Number of VFs per PF** – Configures the number of PCI Virtual Functions Advertised by the port in PCI config space when SR-IOV is enabled in non-NPAR mode. This setting is available for configuration only in non-NPAR mode. This is a numeric setting. The value must be specified in multiples of 8.

  Note: The maximum number of VFs supported by software is 128 VFs per device.
- **SR-IOV –** Configures Single Root - I/O Virtualization (SR-IOV) which allows different virtual machines (VMs) in a virtual environment to share a single PCI Express hardware interface.
  - **Enabled** – Enable support for SR-IOV
  - **Disabled** – Disable support for SR-IOV
- **Number of MSI-X Vectors per VF**

  Configures the MSI-X Vectors per VF. Message Signaled Interrupts (MSI) are an alternative in-band method of signaling an interrupt using special in-band messages to replace traditional out-of-band assertion of dedicated interrupt lines. This is a numeric setting. The maximum number of virtual functions supported by the adapter is shared across the number of physical ports on the adapter. Keep the default value of 16 for the BCM574XX and 8 for the BCM5750X to achieve the best resource allocation.

  Note: To support the maximum number of VFs per PF when NPAR_EP or RDMA is enabled with SRIOV, set the MSI-X Vectors per VF value to 4.
- **Maximum Number of PF MSI-X Vectors** – Configures the Maximum Number of MSI-X Vectors for a physical function. This is a numeric setting. The minimum value for this setting is 1. The maximum value varies across adapters. Keep the default value of 74 to achieve the best resource allocation and maximum number of VFs.
- **Energy Efficient Ethernet** – Configures the Energy Efficient Ethernet (EEE) mode which is a set of enhancements that allow for less power consumption during periods of low-data activity. This setting is available only on 10GBASE-T controllers.
  - **Enabled** – Turn on EEE mode
  - **Disabled** –Turn off EEE mode
- **Support RDMA** – Configures Remote Direct Memory Access (RDMA) support on the port. RDMA is a technology that permits computers on a network to exchange data in main memory without the involvement of the processor, cache, or operating system of either computer. RDMA allows high throughput and low-latency networking. This setting is available only when RDMA is supported on the adapter. This setting will be displayed on the Device Configuration menu in SF mode and the NIC Partition Configuration menu in NPAR mode.
  - **Enabled** – Turn on RDMA
  - **Disabled** – Turn off RDMA
- **Support RDMA on VFs** – Configures RoCE support for the child VFs on the physical Function. It is only valid when both RDMA and SR-IOV are enabled. This setting is ignored when RDMA support on the PF is disabled. This feature is not supported in NPAR mode. This setting is only available on the BCM95750X and BCM957608 family of Ethernet network adapters.
  - **Enabled** – Turn on RDMA support on VFs
  - **Disabled** – Turn off RDMA support on VFs

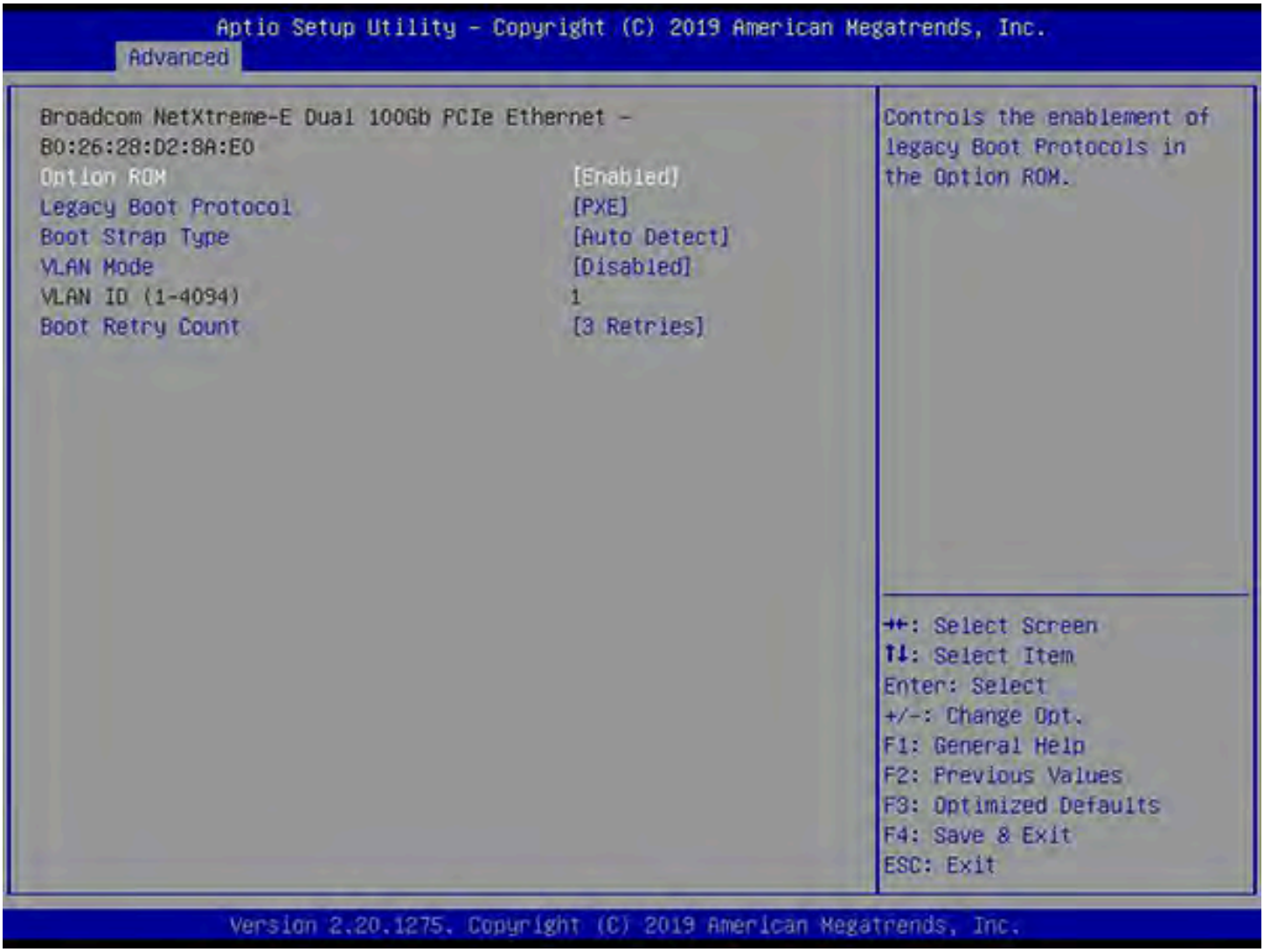
- **DCB Protocol** – Configures the Data Center Bridging (DCB) settings for the controller. Some of the following options may not be available depending on the adapter in use.
  - **Enabled (IEEE only)**
  - **CEE (only)**
  - **Both (IEEE preferred with fallback to CEE)**
- **LLDP Nearest bridge** – Configures the Link Layer Discovery Protocol (LLDP) which is a vendor-neutral link layer protocol used by network devices for advertising their identity, capabilities, and neighbors on a local area network based on IEEE 802 technology, principally wired Ethernet. An LLDP agent is a mapping of an entity where LLDP runs.
  - **Enabled** – Turn on LLDP nearest bridge
  - **Disabled** – Turn off LLDP nearest bridge
- **LLDP nearest non-TPMR bridge** – Configures the Link Layer Discovery Protocol (LLDP) which is a vendor-neutral link layer protocol used by network devices for advertising their identity, capabilities, and neighbors on a local area network based on IEEE 802 technology, principally wired Ethernet. An LLDP agent is a mapping of an entity where LLDP runs. This setting enables LLDP on the nearest non-TPMR bridge agent.
  - **Enabled** – Turn on LLDP nearest non-TPMRbridge
  - **Disabled** – Turn off LLDP nearest non-TPMRbridge
- **Default EVB Mode** – Configures the Edge Virtual Bridging (EVB) mode which is an IEEE standard that involves the interaction between virtual switching environments in a hypervisor and the first layer of the physical switching infrastructure.
  - **VEB**
  - **VEPA**
  - **None**
- **Enable PME Capability** – Configures PME which is the ability to remotely wake a server using Power Management Event (PME). Devices supporting the native PCI Power Management can generate wakeup signals called PMEs to let the kernel know about external events requiring the device to be active.
  - **Enabled** – Turn on PME.
  - **Disabled** – Turn off PME.
- **Flow Offload** – Configures Flow Offload Mode. This feature is supported on Linux only. It is not supported on Windows or ESX OS.
  - **Enabled** – Turn on Flow Offload Mode.
  - **Disabled** – Turn off Flow Offload Mode.
- **Port Enablement** – Configures the number of active ports (PCI functions) on the adapter. This setting is available only on adapters that support the Port Enablement feature.
  - **Enable all ports**
  - **Disable ports 2, 3, and 4** – Available only on quad-port adapters.
  - **Disable ports 3 and 4** – Available only on quad-port adapters.
  - **Disable port 2 or 4** – Displayed as 2 on dual-port adapters and as 4 on quad-port adapters.
- **Adapter Error Recovery** – Enables the recovery of firmware from fatal errors without manual intervention, host reboot, or power cycle.
  - **Enabled** – Turn on Adapter Error Recovery
  - **Disabled** – Turn off Adapter Error Recovery
- **BAR2 Size** – Configures the size of PCI BAR2 space. This option impacts the amount of doorbell space available. This setting is only available on the BCM95750X and BCM957608 family of Ethernet network adapters. Possible values are: Disabled, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, 64M, 128M, 256M, 512M, and 1G.
- **Performance Profile** (BCM95750X/BCM957608 adapters only) – Configures the performance profile that identifies a set of parameters to configure the hardware for the optimized performance.
  - **Default** – Performance is optimized for IP (L2) traffic.
  - **RoCE** – Performance is optimized for RoCE traffic, while supporting IP traffic as lower priority.

MBA Configuration Menu NIC Configuration

Note: MBA Configuration is not available for BCM957608 devices.

This menu presents legacy boot-related parameters for configuration. Depending on the adapter type, some settings may not be available.

Figure 5: MBA Configuration Menu

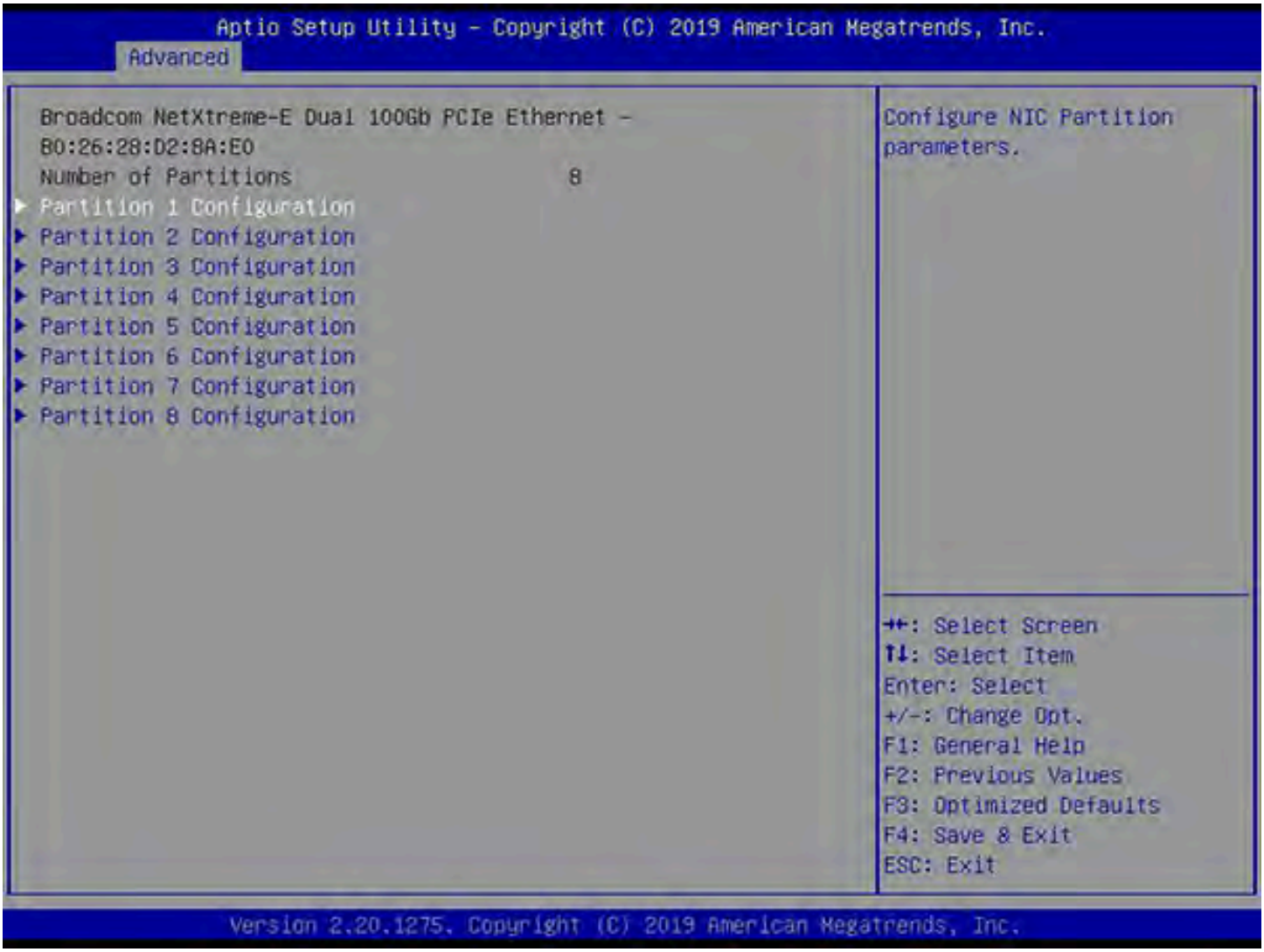


The MBA configuration/NIC Configuration menu consists of the following items:

- **Option ROM** – Allows the user to control whether the Broadcom legacy option ROM driver should be advertised or not.
  - **Enabled** – Load legacy option ROM driver
  - **Disabled** – Do not load legacy option ROM driver
- **Legacy Boot Protocol** – Configures the type of boot to be performed in legacy mode.
  - **PXE**
  - **None**
- **Boot Strap Type** – Configures the bootstrap protocol for legacy PXE boot.
  - **Auto Detect**
  - **BBS**
  - **Int 18h**
  - **Int 19h**
- **Pre-boot Wake on LAN** – Configures Wake on LAN (WoL) which is the ability to remotely power on a server or to wake it up from sleep mode. This setting is available only on adapters that support the **Wake on LAN** feature.
  - **Enabled** – Turn on WoL
  - **Disabled** – Turn off WoL
- **VLAN Mode** – Configures a virtual LAN.
  - **Enabled** – Turn on VLAN mode
  - **Disabled** – Turn off VLAN mode
- **VLAN ID** – Configures a VLAN tag when VLAN mode is enabled. This is a numeric setting. The value must be specified in the range of 1 to 4094.
- **Boot Retry Count** – Configures the number of times legacy boot must be attempted in case of failure.
  - No Retry
  - 1 Retry
  - 2 Retries
  - 3 Retries
  - 4 Retries
  - 5 Retries
  - 6 Retries
  - Indefinite Retries
- **Permit Total Port Shutdown** – This feature is supported on Linux OS only. This setting allows the port to be completely disabled when a port down command is received from the host OS or driver. This feature is not supported when virtualization mode is NPAR or NPAR + SRIOV.
  - **Enabled** – Permit Total Port Shutdown
  - **Disabled** – Permit Total Port Shutdown

NIC Partitioning Configuration Menu

Figure 6: NIC Partitioning Configuration Menu



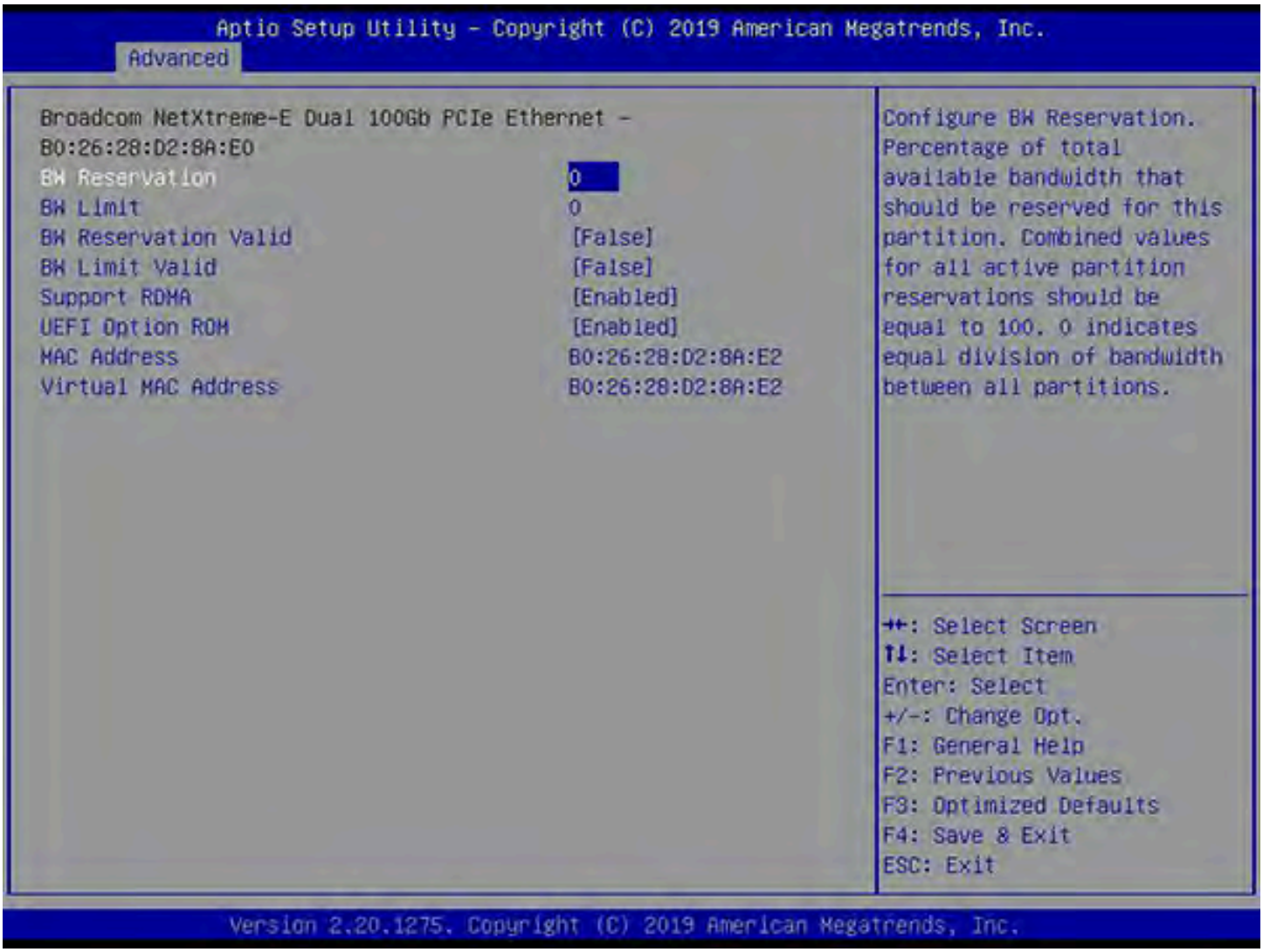
The **NIC Partitioning Configuration** screen has the following sub-menus:

- **Number of Partitions Per Port** – This field displays the number of PCI Physical functions currently enabled on the current network port when MF mode is set to NPAR. This is a read-only field.
- **Partition <n>Configuration** – This menu presents configuration parameters for the nth partition on the network port in NPAR mode. The number of menus presented depends on the value of the **Number of Partitions** field.

Partition n Configuration Menu

This menu presents partition-related parameters for configuration. Depending on the adapter type, some settings may not be available.

Figure 7: Partition n Configuration Menu



- **BW Reservation (Not supported on BCM95750X/BCM957608 devices)** – Configures the percentage of total available bandwidth that should be reserved for this partition. The total Bandwidth Reservation assigned for all active partitions cannot exceed 100. A value of 0 on all partitions indicates an equal division of bandwidth between all partitions. This setting is available only on adapters that support the Bandwidth Reservation feature. This is a numeric setting. The value must be specified in the range of 0 to 100.
- **BW Limit** – Configures the maximum percentage of available bandwidth this partition is allowed. This is a numeric setting. The value must be specified in the range of 0 to 100.
- **BW Reservation Valid** – Configures whether BW Reservation is applicable on the current partition. When this setting is disabled, the BW Reservation value for the current partition will be ignored. This setting is available only on adapters that support the Bandwidth Reservation feature.
  - **Enabled** – Turn on BW Reservation
  - **Disabled** – Turn off BW Reservation
- **BW Limit Valid** – Configures whether BW Limit is applicable on the current partition. When this setting is disabled, the BW Limit value for the current partition will be ignored.
  - **Enabled** – Turn on BW Limit
  - **Disabled** – Turn off BW Limit

- **Support RDMA** – Configures the RoCE support for the current partition. This setting is available only when RDMA is supported on the current partition. This setting will be displayed on the Device Configuration menu in SF mode and in the **NIC Partition Configuration** menu in NPAR mode.
  - **Enabled** – Turn on RDMA
  - **Disabled** – Turn off RDMA
- **UEFI Option ROM** – Configures the setting for advertising the Option ROM during boot time. This setting, if disabled, will not advertise the UEFI Option ROM on that PCI function.
  - **Enabled** – Turn on Option ROM advertisement
  - **Disabled** – Turn off Option ROM advertisement

  Note: This option will not appear on the first partition.
- **MAC Address** – Displays the Permanent MAC address assigned during manufacturing for the current partition. This is a read-only field.
- **Virtual MAC Address** – Displays the Virtual MAC address assigned to the current partition. This is a read-only field from the HII menu. The value for this parameter can be configured using remote utilities.

Link Configuration Menu

This menu presents link configuration parameters.

Figure 8: Link Configuration Menu

Autodetect Speed Exclude Mask – This setting excludes one or more link speeds from the auto-detect link mechanism. Setting a bit in this mask ensures that the specified speed is not attempted by the autodetect state machine. This setting is only available on the BCM5750X/BCM57608 family of Ethernet network adapters. This setting value must be specified in hex. The bitmasks for the flags are as follows - 1G(1), 10G(2), 25G(4), 40G(8), 50G(10), 100G(20), 200G(40), 50G_PAM4(80), 100G_PAM4(100), 100G_PAM4_112(200), 200G_PAM4_112(400), 400G_PAM4(800), 400G_PAM4_112(1000), 2_5G(4000), 100M(8000).

Operational Link Speed – Configures the default link speed for pre-OS environment in full-power (D0) state. The possible values for this setting depend on the link speeds supported by the adapter.

Table 9: Supported Link Speed and Device Support

Link Speed	Device Support	NRZ, PAM4_56, PAM4_112
400G over 4 lanes	BCM957608	PAM4_112
400G over 8 lanes	BCM957608	PAM4_56

Link Speed	Device Support	NRZ, PAM4_56, PAM4_112
200G over 2 lanes	BCM957608	PAM4_112
200G over 4 lanes	BCM957508, BCM957608	PAM4_56
100G over 1 lane	BCM957608	PAM4_112
100G over 2 lanes	BCM957508, BCM957608	PAM4_56
100G over 4 lanes	BCM95750X, BCM957608	NRZ
50G over 1 lane	BCM95750X, BCM957608	PAM4_56
50G over 2 lanes	BCM95741X, BCM95750X, BCM957608	NRZ
25G over 1 lane	BCM95741X, BCM95750X, BCM957608	NRZ

Note: The value for this setting is fixed on some adapters based on the configuration.

- **Media Auto Detect** – Configures the Media Auto Detect feature. This setting is not available on 10GBASE-T controllers.
  - **Enabled** – Turn on Media Auto Detect.
  - **Disabled** – Turn off Media Auto Detect.
- **Auto-negotiation Protocol** – Configures the Auto-negotiation protocol for the adapter. Auto-negotiation is a feature that allows a port on a switch, router, server, or other device to communicate with the device on the other end of the link to determine the optimal duplex mode and speed for the connection. This setting is not available on 10GBASE-T controllers.
  - **IEEE and BAM**
  - **IEEE and Consortium**
  - **BAM Only**
  - **Consortium Only**
  - **IEEE 802.3by**

Link FEC – Configures the Forward Error Correction (FEC) mode which is a technique used for controlling errors in data transmission over unreliable or noisy communication channels. This option is useful when longer fiber cables are utilized. This setting is not available on 10GBASE-T controllers. Only a subset of the possible values displays on some adapters, based on the configuration. Possible values are:

Table 10: Link FEC Modes and Devices

FEC Modes	Devices
CL74 – Fire Code	BCM95741X, BCM95750X (For BCM95750X devices, FEC CL74 mode is supported for 25G link speeds only.)
CL91 – Reed Solomon	BCM957416, BCM95750X, BCM957608 (FEC CL91 mode is supported for 25G NRZ, 50G NRZ, and 100G NRZ speeds only.)
RS544 – RS544, using 1 x N RS	BCM95750X, BCM957608 (FEC RS544_1xN mode is supported for 50G PAM-4 and 100G PAM-4 speeds only.)
RS272 – RS272, using 1 x N RS	BCM95750X
RS544 – RS544, using 2 x N RS	BCM95750X, BCM95760X (FEC RS544_2xN mode is supported for 200G PAM4 speed and up.)
RS272 – RS272, using 2 x N RS	BCM95750X

Note: When Media Auto Detect and Auto-negotiation are enabled, FEC is negotiated based on the link partner advertisement.

Port Link Training – Configures Port Link Training when using forced link speed. See the following table for when to enable or disable link training depending on the cable type.

- **Enabled** – Turn on Port Link Training
- **Disabled** – Turn off Port Link Training

Table 11: Link Training Configuration per Cable Type

Cable Type	Link Training Enable/Disable
Direct Attach Copper (DAC)	Link Training – Enabled
Active Optical Cable (AOC)	Link Training – Disabled
Active Electrical Cable (AEC)	Link Training – Disabled
Active Copper Cable (ACC)	Link Training – Enabled
Linear Drive Pluggable Optics (LPO)	Link Training – Disabled

Installing Software for Ethernet Network Adapters

Provides instructions for installing the driver software for Ethernet network adapters under Linux, VMware, and Windows.

Software is comprised of the driver, firmware, library, and utility components. All components are bundled in files available from the Broadcom Ethernet network adapters site under the **Downloads** > **Driver** section located on each device page.

The Software Installation section contains the following sections:

- Supported Operating Systems for Ethernet Network Adapters
- Installing the Linux Driver on Ethernet Network Adapters
- Installing the VMware Driver on Ethernet Network Adapters
- Installing the Windows Driver on Ethernet Network Adapters
- Updating the Firmware on Ethernet Network Adapters

Supported Operating Systems for Ethernet Network Adapters

Provides a list of supported operating systems for Ethernet network adapters.

The following tables provide a list of supported operating systems for BCM95741X/BCM5750X and BCM957608 devices.

Table 12: BCM95741X/BCM95750X/BCM957608 Supported Operating Systems

Operating System	Distribution	Binary Packages	Source Packages
Debian	10.13 (BCM957608 only), 12.9, 12.11, and 13	None	Yes
DPDK	19.11 (BCM95741X/BCM95750X only), 22.11, 23.11, and 24.11	None	Yes
FreeBSD	14.3	None	Yes
OpenEuler OS	22.03 LTS SP3, 22.03 LTS SP4, 24.03 LTS SP1, and 24.03 LTS SP2	None	Yes
Oracle (Inbox Drivers Only)	UEK R8	N/A	N/A
RHEL	9.5, 9.6, 9.7 (Inbox only), 10.0, 10.1 (Inbox only)	x86	Yes
	9.5, 9.6, 9.7 (Inbox only), 9.8 (Inbox only), 10.0, 10.1 (Inbox only), and 10.2 (Inbox only)	x86	Yes
SuSE SLES	12 SP5, 15 SP6, 15 SP7, and 16 (Inbox only)	x86	Yes
Ubuntu	22.04.4 LTS, 22.04.5 LTS, 24.04.2 LTS, and 24.04.3 LTS	x86	Yes
	22.04.4 LTS, 22.04.5 LTS, 24.04.2 LTS, and 24.04.3 LTS	x86	Yes
vSphere/ESX	8.0 u2, 8.0 u3, and 9.0	x86	N/A
Windows	Windows 11 23H2 Win PE and Windows 11 24H2	x86	N/A
Windows Server	2022 and 2025	x86	N/A

Table 13: DOCA Support for BCM95750X/BCM957608

Operating Systems	2.8	2.9	2.10
Ubuntu	22.04 and 24.04	22.04 and 24.04	22.04 and 24.04
RHEL	8.9, 8.10, and 9.4	8.9, 8.10, and 9.4	8.9, 8.10, 9.4, and 9.5
SLES	15 SP5 and 15 SP6	15 SP5 and 15 SP6	15 SP5 and 15 SP6

Table 14: x86 MOFED Support with BCM95750X/BCM957608

Operating Systems	MOFED 24.04	MOFED 24.07	MOFED 24.10
Ubuntu	22.04 and 24.04	22.04 and 24.04	22.04 and 24.04
RHEL	8.9 and 9.4	8.9 and 9.4	8.9, 9.4, and 9.5
SLES	15 SP5	15 SP5 and 15 SP6	15 SP5 and 15 SP6

Installing the Linux Driver on Ethernet Network Adapters

Provides information on installing the Linux driver (manually or automatically) as well as the L2 and RoCE drivers.

- Installing the L2 and RoCE Drivers Using the Automated Installer
- Installing and Configuring the Software Manually
- Useful Linux Commands

Installing the L2 and RoCE Drivers Automatically

Provides instructions for installing the Linux driver (manually or automatically). L2 as well as the RoCE drivers.

The installation package includes an automation tool for software installation and system configuration. Using the installation package is the preferred installation method.

This section is based on a ZIP file (Broadcom Ethernet network adapter Linux Driver Installer) which is extracted to a directory referenced as follows as `$REL_DIR`.

Note: For RoCE-specific software information and prerequisites, see RoCE Software Specific Information.

The installer provides the following:

1. Installs the included drivers, firmware, RDMA library, and utility tools.
2. Configures interface priorities for IP and RoCE.
3. Provides switch configuration examples.

The following additional system packages are automatically installed as needed:

- libibverbs-utils, rdmacm-utils, perftest
- gcc, make, rpmbuild, kernel headers: if necessary to compile drivers or library

SuSE Linux

For SuSE systems with Secure Boot enabled, refer to the steps in the following link to import the Partner Linux Driver Program key:

https://drivers.suse.com/doc/Usage/Secure_Boot_Certificate.html

For Broadcom Partner Linux Driver Program certificates, see the `$REL_DIR/drivers_linux` folder for `.der` certificate files and additional instructions in the `readme_secure_boot.txt` file.

Installing the Drivers Automatically

The Broadcom installation package comes with an installer that can be used to install and configure RoCE drivers and libraries as well as host dependencies. The use of the installer is the recommended approach for installing RoCE on a new system.

The installer performs the following steps:

- Installs the kernel L2 and RoCE drivers
- Installs the user-space RoCE library
- Installs other host packages needed for RoCE (`libibverbs` and so forth)
- Installs the required firmware
- Installs the Broadcom `niccli` , `bcm_sosreport` , and `bnxt_re_conf` utilities
- Applies the required firmware configuration for RoCE
- Applies the required configuration for RoCE congestion control

Use the following commands to start the automated installation with default values and all installer components for an interface that is running and has an IP address:

Note: Root privileges are required to run the `install.sh` command.

```
cd $REL_DIR/utils/linux_installer
sudo bash install.sh -i <IFACE>
```

Note: <IFACE> must be replaced with the interface name or the device's PCIe address: For example, p1p1 or 41:00.0

Use the following commands to install only the L2/Ethernet driver without installing/configuring RoCE:

```
cd $REL_DIR/utils/linux_installer
sudo bash install.sh -i <IFACE> -2
```

Use the following commands to start the automated installation with IP address and MTU specified with verbose output:

```
cd $REL_DIR/utils/linux_installer
sudo bash install.sh -v -i <IFACE> -a <IP> -n <NETMASK> -m <MTU>
```

For a complete explanation of the automated installer including all options and modes of use, refer to the README file in `$REL_DIR/utils/linux_installer`.

Note: The automated installer installs and updates both the L2 and RoCE drivers. To use the RoCE feature, see RoCE Configuration for additional configuration details.

RoCE-Specific Software Information

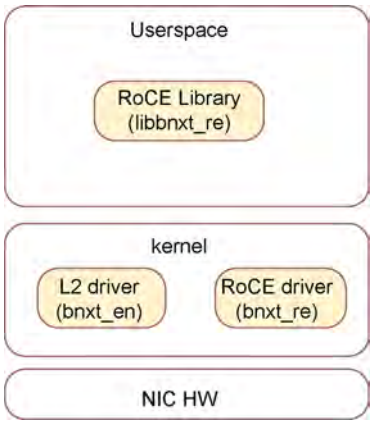
Although the RoCE software is installed as part of the L2 driver installation, there is general information and prerequisites that are specific to RoCE.

Linux Software Components

The Linux RoCE software consists of the following components:

- Broadcom L2 Ethernet Driver (`bnxt_en`)
- Broadcom RoCE Driver (`bnxt_re`)
- Broadcom RoCE user-space library, also known as RoCE user-space driver (`libbnxt_re`)
- Broadcom Utility Tools to configure RoCE on the NIC (`bnxt_re_conf`)

Figure 9: RoCE Software Components



All the SW components are part of an installation package available from the Ethernet Network Adapters page under the downloads section of each Ethernet adapter. For example, for the P2100G adapter, the software is available from the **Downloads** section of the P2100G - 2 x 100GbE PCIe NIC.

Linux Software Installation Prerequisites

The Broadcom RoCE implementation uses the Linux libibverbs RDMA software stack (`librdma libibverbs`) included with all major Linux distributions.

Note: Any installation of another vendor's proprietary software stack should be removed before proceeding with these instructions.

Note: If using the automated software Installation using the Broadcom installer, these dependencies are automatically downloaded and installed into the system.

Linux: Details of Automated SW Installation for RoCE

The automated Linux installer comes with a `readme.txt` file that provides details on the workings of the installer, as well as all the installation options available with the installer. The installer also has a built-in help option that lists all the available installation options.

```
cd $REL_DIR/utils/linux_installer
sudo bash install.sh -h

Usage: install.sh [-h] [-v] [--uninstall] [-i DEVICES [DEVICES ...]] [-A]
[-a IP_ADDRESS [IP_ADDRESS ...]] [-n NETMASK [NETMASK ...]]
[-l IP_ADDRESS_PREFIX] [-m MTU] [-w] [-s RELEASE_SOURCE]
[-o {ECN,PFC,ECNPFC,NOOP}] [-r ROCE_PRI] [-d ROCE_DSCP]
[-c CNP_PRI] [-p CNP_DSCP] [-b ROCE_PCT] [-2] [-q VLAN] [-g]
[-f] [-t] [-k] [-e] [-0] [--minimal] [-R] [-S]

Optional Arguments:
-h, --help: Shows this help message and exit
-v, --verbose: Shows all commands run and status information
--uninstall: Uninstalls packages and boot-time configuration of the specified devices (-d)
-i DEVICES [DEVICES ...] Specifies a network interface. PCI addresses or interface names (
eno1np0,eno2np1 )are allowed.
-A: Applies on all supported network interfaces. All interface names ( eno1np0,eno2np1 )are selected.
-a IP_ADDRESS [IP_ADDRESS ...]: Specifies an IP address for the previous interface
-n NETMASK [NETMASK ...]: Specifies the netmask for the previous interface. Default: 255.255.255.0.
syntax: -a <address> -n <netmask>
-l IP_ADDRESS_PREFIX: Configures interface IP address as <prefix>.(last octet of public interface)
```

```
-m MTU, --mtu: MTU Interface MTU. Default: 1500. syntax: -a <address> -m <MTU>
-w, --no-firmware: Do not install firmware
-s RELEASE_SOURCE, --source RELEASE_SOURCE: Path to release files. Default: ../
-o {ECN,PFC,ECNPFC,NOOP}, --cc_mode {ECN,PFC,ECNPFC,NOOP}: RoCE congestion control mode. Default: NOOP
-r ROCE_PRI, --roce_pri ROCE_PRI: RoCE priority. Must set -o to take effect. Default: 3
-d ROCE_DSCP, --roce_dscp ROCE_DSCP: RoCE Packet DSCP value. Must set -o to take effect. Default: 26
-c CNP_PRI, --cnp_pri CNP_PRI: RoCE CNP Packet Priority. Must set -o to take effect. Default: 7
-p CNP_DSCP, --cnp_dscp CNP_DSCP: RoCE CNP DSCP value. Must set -o to take effect. Default: 48
-b ROCE_PCT, --roce_pct ROCE_PCT: RoCE Bandwidth percentage for ETS configuration. Must set -o to take
effect. Default: 90
-2, --l2: Do not install/configure RoCE. Install IP driver only
-q VLAN, --vlan VLAN: Specifies VLAN tag to use instead of DSCP for RoCE
-g, --peermem: Enables peer-mem capable drivers -f, --force Force installation, even if it is a
owngrade, or if versions already match
-t, --nictune: Runs nictune tool to tune the specified interface for optimal throughput
-k, --bcm_sosreport: Runs bcm_sosreport tool to collect system information for support purposes
```

Installing and Configuring the Software Manually

Provides instructions for installing and configuring the software manually.

The automated installer is the preferred method to install the L2 and RoCE drivers, however, the installation can also be performed manually.

This section is based on a ZIP file (Broadcom Ethernet network adapter Linux Installer) which is extracted to a directory referenced as follows as `$REL_DIR`

The requirements to set up Linux RoCE and Linux RoCE congestion control are described in the following sections.

Installing Prerequisite Packages

Use the Linux distribution's package manager to install prerequisites for compiling and using RDMA devices. To update the firmware manually, see Updating the Firmware Manually on Linux/ESX.

Note: It is recommended to use the latest version of the Linux distribution.

SuSE Linux

For SuSE systems with Secure Boot enabled, refer to the steps in the following link to import the Partner Linux Driver Program key:

https://drivers.suse.com/doc/Usage/Secure_Boot_Certificate.html

For Broadcom Partner Linux Driver Program certificates, see the `$REL_DIR/drivers_linux/` folder of the driver download ZIP files for `.der` certificate files and additional instructions in the `readme_secure_boot.txt` file.

Debian/Ubuntu

```
sudo apt install -y automake autoconf libtool libibverbs-dev ibverbs-utils infiniband-diags perftest ethtool
```

Red Hat/Fedora/CentOS

```
sudo yum install -y libibverbs-devel qperf perftest infiniband-diags make gcc kernel kernel-devel autoconf
 aclocal libtool libibverbs-utils rdma-core-devel
```

Building and Installing IP and RoCE Drivers

Broadcom's Ethernet device driver (`bnxt_en`) provides an Ethernet interface and Broadcom's RDMA driver (`bnxt_re`) provides the RoCE interfaces. Load both `bnxt_en` and `bnxt_re` to use the RDMA capability of Broadcom RNICs.

The bnxt_en and bnxt_re modules must come from the same package. Extract, compile, and install both drivers using the commands in the following sections.

Enable RDMA

Some Broadcom RNICs have RDMA disabled in the default factory configuration to provide the maximum resources possible to IP and DPDK applications. Ensure that the NIC has RDMA enabled by using the following commands:

NICCLI Command

```
niccli -i <index> nvm --setoption support_rdma --scope <function number> --value 1
niccli -i <index> nvm --setoption performance_profile --value 1
sudo reboot
```

Option 1: Installing from the Kernel RPM

```
cd $REL_DIR/driver_linux/bundle/kmp/<distro>/<distro-version>
sudo rpm -i kmod-bnxt_en-x.y.z.<distro-version.x86_64>.rpm
Repeat the same step above for the bnxt_re driver
```

Option 2: Compiling and Installing from the Source RPM

```
cd $REL_DIR/drivers_linux/bundle/kmp/<distro>/<distro-version>
sudo rpmbuild --rebuild bnxt_en-x.y.z.src.rpm
sudo rpm -i /root/rpmbuild/RPMS/x86_64/kmod-bnxt_en-x.y.z.rpm
sudo depmod -a
sudo modprobe bnxt_en
sudo modprobe bnxt_re
```

Option 3: Compile and Install Directly from Source

```
cd $REL_DIR/drivers_linux/bundle
tar xf netxtreme-bnxt_en-x.y.z.tar.gz
cd netxtreme-bnxt_en-x.y.z
make
sudo make install
sudo depmod -a
sudo modprobe bnxt_en
sudo modprobe bnxt_re
```

Removing the Linux Driver

Note: The following commands to remove the Linux drivers are supported in software revision 2.27 or later.

Note: The bnxt_re driver should not be unloaded from a PF while VFs are using RoCE.

The following command removes and unloads both the bnxt_re and bnxt_en drivers:

```
modprobe -r bnxt_re
```

To remove only the RoCE driver, use the following command:

```
rmmod bnxt_re
```

Updating Initramfs

Most Linux distributions use a ramdisk image to store drivers for boot-up. These kernel modules will take precedence, so the initramfs must be updated after installing the new bnxt_en /bnxt_re modules:

Debian/Ubuntu

```
sudo update-initramfs -u
```

Red Hat/CentOS/Fedora

```
sudo dracut -f
```

IP Configuration

To configure the NIC, first determine the interface name by using the following command:

```
dmesg | grep "$(lspci |grep BCM575 | cut -d' ' -f1)" |grep NIC
```

This command identifies the NIC interface name, connection speed, and PCIe bus:device:function.

Creating a Boot-Time Configuration File Debian/Ubuntu

Add to /etc/network/interfaces:

```
auto <iface>
iface <iface> inet static
address w.x.y.z
netmask a.b.c.d
gateway e.f.g.h
```

Note: <iface> must be in the form of <iface>. Use the <vlan id> to use VLANs.

Red Hat/Fedora/CentOS

```
Edit /etc/sysconfig/network-scripts/ifcfg-<iface>:
DEVICE=<iface>
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
NETMASK=a.b.c.d
IPADDR=w.x.y.z
# Uncomment below for VLAN
#VLAN=yes
#VLAN_EGRESS_PRIORITY_MAP=0:0,1:1,2:1,3:1,4:1,5:1,6:1,7:1
```

Note: <iface> must be in the form of <iface>.Use <vlan id> to use VLANs.

Bring Up Interface

After the interface is defined in the previous steps, bring up the interface as:

```
sudo ifup <IFACE>
```

User Space RDMA Library

The `libbnxt_re library` provides the RDMA verbs interface to applications.

Removing the Conflicting Library File from the Linux Distribution

```
sudo (find /usr/lib64 -name libbnxt_re-rdmav\*.so; find /usr/lib -name libbnxt_re-rdmav\*.so) | xargs
-i{} mv {} {}.bak
```

Option 1: Installing from Binary RPM

Installing from RPM is the preferred method.

```
rpm -i $REL_DIR/drivers_linux/bnxt_rocelib/rpm/<distro>/<distro version>/libbnxt_re-x.y.z.rpm
```

Option 2: Compiling and Installing from Source RPM

To install from a source RPM, use the following commands:

```
rpmbuild --rebuild $REL_DIR/drivers_linux/bnxt_rocelib/rpm/<distro>/<distro version>/libbnxt_rex.
y.z.src.rpm
sudo rpm -i /root/rpmbuild/RPMS/x86_64/libbnxt_re-x.y.z.rpm
```

Option 3: Compiling and Installing from Source

The `libbnxt_re` library can also be installed directly from the source.

```
cd $REL_DIR/drivers_linux/bnxt_rocelib
tar xf libbnxt_re-x.y.z.tar.gz
cd libbnxt_re-x.y.z
sh autogen.sh
./configure --sysconfdir=/etc
make
make install all
sudo sh -c "echo /usr/local/lib >> /etc/ld.so.conf"
sudo ldconfig
cp bnxt_re.driver /etc/libibverbs.d/
```

Verifying the RoCE Library Installation

To verify the RoCE library installation, use the following command:

```
# ibv_devices
device node GUID
------ ----------------
bnxt_re0 6e92cffffe1cc320
bnxt_re1 6e92cffffe1cc321
```

If a library is not replaced properly, the following warning displays:

```
ibv_devices
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 8 (supports 1 to 1) for device /sys/
class/infiniband/bnxt_re1
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 8 (supports 1 to 1) for device /sys/
class/infiniband/bnxt_re1
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 8 (supports 1 to 1) for device /sys/
class/infiniband/bnxt_re0
libibverbs: Warning: Driver bnxt_re does not support the kernel ABI of 8 (supports 1 to 1) for device /sys/
class/infiniband/bnxt_re0
device node GUID
```

Installing the bnxt_re_conf Scripts

This section provides information on installing the `bnxt_re_conf` .

Option 1: Installing the bnxt_re_conf scripts from a Binary RPM

To install `bnxt_re_conf` from a binary RPM, use the following command:

```
rpm -i $REL_DIR/drivers_linux/ bnxt_re/bnxt_re_conf/bnxt_re_conf-x.y.z.noarch.rpm
```

Option 2: Installing the bnxt_re_conf scripts from the Source

To install `bnxt_re_conf` from the source, use the following command:

```
cp bnxt_re.conf /etc/bnxt_re/
cp bnxt_re_conf.sh /usr/bin/
cp bnxt_setupcc.sh /usr/bin/
cp 90-bnxt_re.rules /usr/lib/udev/rules.d/
```

Configuring User Memory Limits

Non-root users typically need access to larger than usual amounts of locked memory, so the system defaults must be updated. Add the following lines to `limits.conf` :

`/etc/security/limits.conf` :

- `soft memlock unlimited`
- `hard memlock unlimited`

Installing the NICCLI Configuration Utility

To install the NICCLI Configuration Utility, see Installing the NICCLI Configuration Utility

Configuring QoS

Configuring QoS can be done with the `niccli` utility. These utilities can be used to set QoS mappings, priority flow control, and configure ETS. These utilities only configure settings on the NIC and require the user to apply a symmetrical configuration on network switches.

These utilities set APPTLVs and configure PFC and ETS:

NICCLI Command

```
niccli [OPTIONS] <COMMAND> <params>
```

Table 15: NICCLI Commands

Command	Description
`-v|--version`	This command displays the program version details.
`qos --ets`	This command configures the priority to TC and bandwidths. This command configures only TSA and Bandwidths.
	Example: `niccli -i 1 qos --ets --tsa 0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict --up2tc 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0 --tcbw 70,30`
`qos --pfc`	This command enables PFC on given priority. Valid values are 0 to 7.
	niccli Syntax: `niccli -i 1 qos --ets --tsa 0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict --up2tc 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0 --tcbw 70,30` with comma separated and in units of percentage (%).
	Example: `niccli -i. 1 qos --pfc --enable 5,6`
`qos --apptlv`	This parameter configures the priority of the APPTLV
	niccli Syntax: `niccli -i <index> qos --apptlv --add --app <priority, selector, protocol>` **niccli Syntax**: `niccli -i <index> qos --apptlv --del --app <priority, selector, protocol>`
	niccli Example: `niccli -i 1 qos --apptlv --add --app 5,1,35093` **niccli Example**: `niccli -i 1 qos --apptlv --del --app 5,1,35093`
`qos --ets --show`	This command gets the configured priorities and bandwidth parameters.
`qos --rate_limit`	This command sets the rate limit for each TC in units of percentage (%).
	niccli Example: `niccli -i 1 qos --tc --set --rate_limit 80,60,70`

Command	Description
`qos --listmap`	This command dumps the supported `dump string` . Supported `dump string` are pri2cos.
	Syntax: `niccli -i 1 qos --listmap --pri2cos`
	Example: `niccli -i 1 qos --listmap --pri2cos`

Tool Changes

The host-side configuration tool provides the following commands to call newly added HWRM APIs for programming and querying ingressqos and egressqos buffer allocation input parameters. The following commands are under niccli as part of the feature parameter configuration.

The following command supports changing the ingress adaptive QoS parameters:

niccli Command

```
niccli -i <index> qos <-n|--ingress> --cosq --set --state <value> [--mode <value>] [-p|--persistent]
```

Table 16: NICCLI Commands

Command	Description
-cosq	This option is used to set the cosq parameter, for example, cosq state.
-state	This is an optional parameter and this field is a bitmask indicating which CoS queues are enabled or disabled. Each bit represents a specific queue where bit 0 represents queue 0 and bit 7 represents queue 7. A value of 0 indicates that the queue is not enabled. A value of 1 indicates that the queue is enabled.
-persistent	If the persistent option is given the configuration is written to NVRAM to save the configuration across reboots.
	niccli Example: `niccli -i 1 qos --ingress --cosq --set --state 255 --mode 16` **niccli Example**: `niccli -i 1 qos --egress --cosq --set --state 255 --persistent`

Note: In the previous examples, all the 8 queues are enabled.

The following command supports changing the egress adaptive QoS parameters:

```
niccli -i <index> qos --ingress --cosq --show --persistent
```

Table 17: NICCLI Commands

Command	Description
-cosq	This option is used to set the cosq parameter, for example, cosq state.
-state	This is an optional parameter and this field is a bitmask indicating which CoS queues are enabled or disabled. Each bit represents a specific queue where bit 0 represents queue 0 and bit 7 represents queue 7. A value of 0 indicates that the queue is not enabled. A value of 1 indicates that the queue is enabled.
-persistent	If the persistent option is given the configuration is written to NVRAM to save the configuration across reboots.
	niccli Example: `niccli -i 1 qos --ingress --cosq --show` **niccli Example**: `niccli -i 1 qos --ingress --cosq --show --persistent`

Note: In the previous examples, all the 8 queues are enabled.

The following command is added to support retrieving ingress adaptive QoS parameters:

```
niccli -i <index> qos --ingress --cosq --show --persistent
```

Table 18: NICCLI Commands

Command	Description
-cosq	This option is used to get the cosq parameter i.e., cosq state.
-persistent	This option is used to query the receive side cosq configuration from the NVRAM.
	niccli Example: `niccli -i 1 qos --ingress --cosq --show` **niccli Example**: `niccli -i 1 qos --ingress --cosq --show --persistent`

	Output:
	`Receive side CoSQ Information :`
	
	`QueueID State Mode`
	`******* ****** *****`
	`0 Enabled Lossy`
	`1 Enabled Lossless`
	`2 Enabled Lossy`
	`3 Enabled Lossy`
	`4 Enabled Lossy`
	`5 Enabled Lossy`
	`6 Enabled Lossy`
	`7 Enabled Lossy`

The following command is added to support retrieving egress adaptive QoS parameters:

```
niccli -i <index> qos <-e|--egress> --cosq --show [-p|--persistent]
```

Table 19: NICCLI Commands

Command	Description
-cosq	This option is used to get the cosq parameter i.e., cosq state.
-persistent	This option is used to query the transmitside cosq configuration from the NVRAM.
	niccli Example: `niccli -i 1 qos --egress --cosq --show --persistent` **niccli Example**: `niccli -i 1 qos --egress --cosq --show`

	Output:
	
	`Transmit side CoSQ Information :`
	`QueueID State`
	`******* ******`
	`0 Enabled`
	`1 Enabled`
	`2 Enabled`
	`3 Disabled`
	`4 Disabled`
	`5 Disabled`
	`6 Disabled`
	`7 Disabled`

QoS Configuration Examples

The `niccli` utility can adjust any RoCE QoS parameters. Use the LLDP agent to use DCBx on your network. For example, the Enhanced Transmission Selection (ETS) ratio can be changed to 50/50 and PFC disabled as follows:

NICCLI Command

```
niccli -i <index> qos --pfc --enable 0xFF
niccli -i <index> qos --ets --tsa 0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict --up2tc
 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0 --tcbw 70,30
```

LLDP Agent

As an alternative to the Broadcom `niccli` tool, the Linux LLDP agent sets QoS mappings and configures ETS to set bandwidth allocation ratios. The LLDP agent also allows the use of DCBx to import remote QoS settings. The LLDP Agent must be installed.

When using lldpagent, ensure that the firmware-based DCBx is disabled with the following command:

NICCLI Command

```
niccli -i <index> nvm --setoption dcbx_mode --scope 0 --value 0
```

Installing the LLDP Agent

Install the LLDP agent and lldptool utility by using the commands in the following sections:

Debian/Ubuntu

```
sudo apt install libconfig9 libnl-3-200
wget http://ftp.us.debian.org/debian/pool/main/l/lldpad/lldpad_0.9.46-3.1_amd64.deb
sudo dpkg -i lldpad_0.9.46-3.1_amd64.deb
sudo systemctl enable lldpad
sudo systemctl start lldpad
```

Red Hat/Fedora/CentOS

```
sudo yum install lldpad
sudo systemctl enable lldpad
sudo systemctl start lldpad
```

NICCLI Utilities

The Broadcom NICCLI utilities allow setting nonvolatile configuration elements of the RNIC, such as enabling or disabling RoCE, SR-IOV, and other options.

Validating the RoCE Interface

Once the RoCE Drivers and the FW have been installed and configured, the `ibstat` command (from the infiniband-diags package) and/or `ibv_devinfo` command (from the ibverbs-utils package) should be run to ensure the installation is correct. If the Ethernet network adapter is connected to a link partner, the port should show active

```
[ ~]# ibstat
CA 'bnxt_re0'
        CA type: Broadcom NetXtreme RoCE HCA
        Number of ports: 1
        Firmware version: 222.0.73.0
        Hardware version: 0x14e4
        Node GUID: 0xbe97e1fffeedfc60
        System image GUID: 0xbe97e1fffeedfc60
```

```
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 50
                Base lid: 0
                LMC: 0
                SM lid: 0
                Capability mask: 0x041d0000
                Port GUID: 0xbe97e1fffeedfc60
                Link layer: Ethernet

[ ~]# ibv_devinfo -vvv
hca_id: bnxt_re0
        transport:                      InfiniBand (0)
        fw_ver:                         222.0.73.0
        node_guid:                      be97:e1ff:feed:fc60
        sys_image_guid:                 be97:e1ff:feed:fc60
        vendor_id:                      0x14e4
        vendor_part_id:                 5968
        hw_ver:                         0x2100
        phys_port_cnt:                  1
        max_mr_size:                    0x8000000000
        page_size_cap:                  0x10201000
        max_qp:                         65537
        max_qp_wr:                      65534
        device_cap_flags:               0x0122dd81
                                        RESIZE_MAX_WR
                                        CURR_QP_STATE_MOD
                                        SHUTDOWN_PORT
                                        PORT_ACTIVE_EVENT
                                        SYS_IMAGE_GUID
                                        RC_RNR_NAK_GEN
                                        N_NOTIFY_CQ
                                        MEM_WINDOW
                                        MEM_MGT_EXTENSIONS
                                        MEM_WINDOW_TYPE_2B
                                        Unknown flags: 0x8000
        max_sge:                        6
        max_sge_rd:                     6
        max_cq:                         131072
        max_cqe:                        8388607
        max_mr:                         262144
        max_pd:                         65536
        max_qp_rd_atom:                 126
        max_ee_rd_atom:                 0
        max_res_rd_atom:                0
        max_qp_init_rd_atom:            126
        max_ee_init_rd_atom:            0
        atomic_cap:                     ATOMIC_GLOB (2)
        max_ee:                         0
        max_rdd:                        0
        max_mw:                         262144
        max_raw_ipv6_qp:                0
```

```
max_raw_ethy_qp:                65536
max_mcast_grp:                  0
max_mcast_qp_attach:            0
max_total_mcast_qp_attach:      0
max_ah:                         131072
max_fmr:                        0
max_srq:                        8192
max_srq_wr:                     1048574
max_srq_sge:                    6
max_pkeys:                      1
local_ca_ack_delay:             16
general_odp_caps:
rc_odp_caps:
                                NO SUPPORT
uc_odp_caps:
                                NO SUPPORT
ud_odp_caps:
                                NO SUPPORT
xrc_odp_caps:
                                NO SUPPORT
completion_timestamp_mask not supported
core clock not supported
device_cap_flags_ex:            0x0
tso_caps:
        max_tso:                        0
rss_caps:
        max_rwq_indirection_tables:                     0
        max_rwq_indirection_table_size:                 0
        rx_hash_function:                               0x0
        rx_hash_fields_mask:                            0x0
max_wq_type_rq:                 0
packet_pacing_caps:
        qp_rate_limit_min:      0kbps
        qp_rate_limit_max:      0kbps
tag matching not supported
        port:   1
                state:                  PORT_ACTIVE (4)
                max_mtu:                4096 (5)
                active_mtu:             1024 (3)
                sm_lid:                 0
                port_lid:               0
                port_lmc:               0x00
                link_layer:             Ethernet
                max_msg_sz:             0x40000000
                port_cap_flags:         0x001d0000
                port_cap_flags2:        0x0000
                max_vl_num:             8 (4)
                bad_pkey_cntr:          0x0
                qkey_viol_cntr:         0x0
                sm_sl:                  0
                pkey_tbl_len:           65535
                gid_tbl_len:            256
                subnet_timeout:         0
```

```
                init_type_reply:        0
                active_width:           1X (1)
                active_speed:           50.0 Gbps (64)
                phys_state:             LINK_UP (5)
        GID[  0]:               fe80:0000:0000:0000:be97:e1ff:feed:fc60, RoCE v1
                GID[  1]:               fe80::be97:e1ff:feed:fc60, RoCE v2

[ ~]#  ibdev2netdev
bnxt_re0 port 1 ==> enp8s0f0np0 (Up)
```

Useful Linux Commands

Provides a list of useful Linux commands and descriptions.

In the following table, `ethX` should be replaced with the actual interface name.

Note: The supported speeds listed by ethtool reflect the controller capability and are not based on the transceiver or cable capability.

Table 20: Linux Ethtool Commands

Command	Description
`ethtool -s ethX speed 25000 autoneg off`	Force the speed to 25G. If the link is up on one port, the driver does not allow the other port to be set to a different speed.
`ethtool -i ethX`	Output includes driver, firmware, and package version.
`ethtool -k ethX`	Show offload features.
`ethtool -K ethX tso off`	Turn off TSO.
`ethtool -K ethX gro off lro off`	Turn off GRO/LRO.
`ethtool -g ethX`	Show ring sizes.
`ethtool -G ethX rx N`	Set ring sizes.
`ethtool -S ethX`	Get statistics.
`ethtool -l ethX`	Show number of rings.
`ethtool -L ethX rx 0 tx 0 combined M`	Set number of rings.
`ethtool -C ethX rx-frames N`	Set interrupt coalescing. Other parameters supported are rx-usecs, rx-frames, rx-usecs-irq, rx-frames-irq, tx-usecs, tx-frames, tx-usecs- irq, tx-frames-irq.
`ethtool -x ethX`	Show RSS flow hash indirection table and RSS key.
`ethtool -s ethX autoneg on speed 10000 duplex full`	Enable Autoneg.
`ethtool --show-eee ethX`	Show EEE state.
`ethtool --set-eee ethX eee off`	Disable EEE.
`ethtool --set-eee ethX eee on tx-lpi off`	Enable EEE, but disable LPI.
`ethtool -L ethX combined 1 rx 0 tx 0`	Disable RSS. Set the combined channels to 1.
`ethtool -K ethX ntuple off`	Disable Accelerated RFS by disabling ntuple filters.
`ethtool -K ethX ntuple on`	Enable Accelerated RFS.
`ethtool -t ethX`	Performs various diagnostic self-tests.

Command	Description
`ethtool -m ethX`	Query/Decode module EEPROM information and optical diagnostics if available.
`echo 32768 > /proc/sys/net/core/rps_sock_flow_entries echo 2048 > /sys/class/net/ethX/queues/rx-X/ rps_flow_cnt`	Enable RFS for ring X.
`sysctl -w net.core.busy_read=50`	This sets the time to read the device's receive ring to 50 µsecs. For socket applications waiting for data to arrive, using this method can decrease latency by 2 or 3 µs typically at the expense of higher CPU utilization.
`echo 4 > /sys/class/net/<NAME>/device/sriov_numvfs`	Enable SR-IOV with four VFs on the named interface.
`ip link set ethX vf 0 mac 00:12:34:56:78:9a`	Set VF MAC address.
`ip link set ethX vf 0 state enable`	Set VF link state for VF 0.
`ip link set ethX vf 0 vlan 100`	Set VF 0 modprobe 8021q; ip link add link <NAME> name <VLAN Interface Name> type vlan id <VLAN ID> **Example:** `modprobe 8021q; ip link add link ens3 name ens3.2 type vlan id 2`

Installing the VMware Driver on Ethernet Network Adapters

Provides information on installing the VMware driver on Ethernet network adapters.

Starting with ESXi version 6.7 and later, Broadcom drivers `bnxtnet` and `bnxtroce` have been combined into a single component bundle and are available as such for download from VMware.com.

1. Use the following command to install the Ethernet and RDMA driver:

   ```
   $ esxcli software component apply -d Broadcom-bnxt-Net-RoCE_<driver version>.zip
   ```

2. Reboot the system for the new driver to take effect.

Other useful VMware commands are shown in the following table.

Note: In the following table, replace `vmnicX` with the actual interface name.

Note: When using NPAR + SR-IOV, the number of VFs that can be enabled is reduced due to resource constraints. When using NPAR + MultiRSS, the driver may create a smaller number of RSS engines due to resource constraints.

Table 21: VMware Commands

Command	Description
`esxcli software vib list |grep bnx`	List the VIBs installed to see whether the bnxt driver installed successfully.
`esxcli software component list |grep bnxt`	List the components installed to see if `bnxt` drivers component installed successfully.
`esxcfg-module –i bnxtnet`	Print module information onto the screen.
`esxcli network get –n vmnicX`	Get vmnicX properties.
`esxcfg-module –g bnxtnet`	Print module parameters.
`esxcfg-module –s 'multi_rx_filters=2 - disable_tpa=0 - max_vfs=0,0 RSS=0,0'`	Set the module parameters.
`vmkload_mod –u bnxtnet`	Unload bnxtnet module.

Command	Description
`vmkload_mod bnxtnet`	Load bnxtnet module.
`kill -SIGHUP $(pidof vmkdevmgr)`	Run after `vmkload_mod` bnxtnet (for loading driver).
`esxcli network nic set -n vmnicX -D full -S 25000`	Set the speed and duplex of vmnicX.
`esxcli network nic down -n vmnicX`	Bring down the vmniX link.
`esxcli network nic up -n vmnic6`	Bring up the vmniX link.

Configuring RoCE in VMware

To configure RoCE in VMware, see VMware Configuration.

Installing the Windows Driver on Ethernet Network Adapters

Provides information on installing the Windows driver on Ethernet network adapters.

Use the following steps to install the Windows drivers:

1. Download and unzip the Windows driver installation package.
2. Click **Server Manager** > **Tools** > **Computer Management** > **Device Manager**.
3. Right-click on the Broadcom devices under **Network Adapters**.
4. Select **Update Driver**.
5. Select **Browse My Computer for driver Device Manager software** and select **Have Disk** to navigate to the folder where the driver files are located.
6. Click **Next** to update the driver automatically.
7. Reboot the system to ensure that the driver is running.

Advanced Properties for Ethernet Network Adapters

Provides information on advanced properties for the Windows driver on Ethernet network adapters.

The Windows driver's advanced properties are shown in the following table.

Table 22: Windows Driver Advanced Properties

Driver Key	Parameters	Description
Encapsulated Task offload	Enable or Disable	Used for configuring NVGRE encapsulated task offload.
Enable PME or Shutdown	Enable or Disable	Some versions of Windows will disable PME (Power Management Enable) for a PCI device when the operating system shuts down. This will prevent the network adapter from being able to wake up the system via magic packet or pattern packet. To override this and allow the network adapter to wake the system, set Enable PME on shutdown to enable.
Encapsulation Overhead	0 to 256	The maximum NVGRE header size from the beginning of the packet to the beginning of the inner packet payload.
Flow control	TX, RX, or TX/RX enable	Configure flow control on RX or TX or both sides.

Driver Key	Parameters	Description
Forward Error Correction	Disabled, CL91 with Force Mode, RS272 1xn Force Mode, RS272 IEEE with Force Mode, and RS544 IEEE with Force Mode	Configure FEC Modes.
Interrupt Moderation	Enable or Disable	Default Enabled. Allows frames to be batch-processed by saving CPU time.
Jumbo packet	1514, 4088, 9014, or 9336	Jumbo packet size.
Large Send offload V2 (IPv4)	Enable or Disable	LSO for IPv4.
Large Send offload V2 (IPv6)	Enable or Disable	LSO for IPv6.
Link Signaling Mode	NZR, PAM-4 112, or PAM-4 56	Configure link signal modulations.
Locally Administered Address	User entered MAC address.	Override default hardware MAC address after operating system boot.
Maximum Number of RSS Processors	Value 16	The maximum number of RSS processors.
Maximum number of RSS Queues	Value 1 to 8	Default is 8. Allows the user to configure Receive Side Scaling queues.
Maximum RSS Processor Number	Value 16	CPUs above this number are not used for RSS.
Network Direct Functionality	Enable or Disable	Enable or disable RDMA over Converged Ethernet (RoCE)
Network Direct MTU	512, 1024, 2048, 4096	Sets the Message Transfer Unit (MTU) for RoCE connections. This setting must match peer settings for RoCE communication.
Network Direct Technology	RoCEv2	Sets the frame format for RoCE connections. This setting must match peer settings for RoCE communication.
NVGRE Encapsulated Task Offload	Enable or Disable	Enable or disable support for Network Virtualization using Generic Routing Encapsulation (NVGRE) Task Offload.
Preferred NUMA node	Value, Not Present	—
Priority and VLAN	Priority and VLAN Disable, Priority enabled, VLAN enabled, Priority and VLAN enabled.	Default Enabled. Used for configuring IEEE 802.1Q and IEEE 802.1P.
PTP Hardware Timestamp	Enable or Disable	Enable or disable Precision Timing Protocol hardware timestamping defined in IEEE 1588.
Quality of Service	Enable or Disable	Enable or disable support for Quality of Service (QoS) for IEEE 802.1 Data Center Bridging (DCB) to specify the policies and settings of traffic classes that the network adapter uses for transmit and priority flow control (PFC).
Receive Buffer (0=Auto)	Increments of 500.	Default is Auto.
Receive Side Scaling	Enable or Disable.	Default Enabled.
Receive Segment Coalescing (IPv4)	Enable or Disable.	Default Enabled.
Receive Segment Coalescing (IPv6)	Enable or Disable.	Default Enabled.

Driver Key	Parameters	Description
RSS Base Processor Group	Value, Not Present	The base processor group for the processor number that is specified in RSS Base Processor Number.
RSS Base Processor Number	Value, Not Present	The number of the base RSS processor in the processor group.
RSS load balancing profile	NUMA scaling static, Closest processor, Closest processor static, conservative scaling, NUMA scaling.	Default NUMA scaling static.
RSS Max Processor Group	Value, Not Present	The maximum processor group.
Software Timestamp	Enable or Disable	Enable or disable Precision Timing Protocol software time stamping as defined in IEEE 1588.
Speed and Duplex	AutoNegotiation, 25 Gbps Full Duplex, 50 Gbps Full Duplex, 100 Gbps Full Duplex, 200 Gbps Full Duplex, or 400 Gbps Full Duplex	Configure link speeds.
SR-IOV	Enable or Disable.	Default Enabled (Disabled for BCM95750X devices). This parameter works with HW-configured SR-IOV and BIOS-configured SR-IOV settings.
TCP/UDP checksum offload IPv4	TX/RX enabled, TX enabled, RX enabled, or offload disabled.	Default RX and TX enabled.
TCP/UDP checksum offload IPv6	TX/RX enabled, TX enabled, RX enabled, or offload disabled.	Default RX and TX enabled.
Transmit Buffers (0=Auto)	Increment of 50.	Default Auto.
VF Spoofing Protection	Enable or Disable.	Enable or disable VF spoofing filter.
Virtual Machine Queue	Enable or Disable.	Default Enabled.
Virtual Switch RSS	Enable or Disable.	Enable or disable RSS for a virtual switch.
VLAN ID	User-configurable number.	Default 0.
VXLAN Encapsulated Task Offload	Enable or Disable.	Enable or disable Virtual Extensible LAN (VXLAN) offload.
Wake on Magic Packet	Enable or Disable.	Enable or disable waking the system when a magic packet is received.
Wake on Pattern Match	Enable or Disable.	Enable or disable waking the system when a packet matching an operating system-defined pattern is received.

Windows Driver RoCE Configuration

For information on configuring the Windows driver for RoCE, see Windows RoCE Configuration.

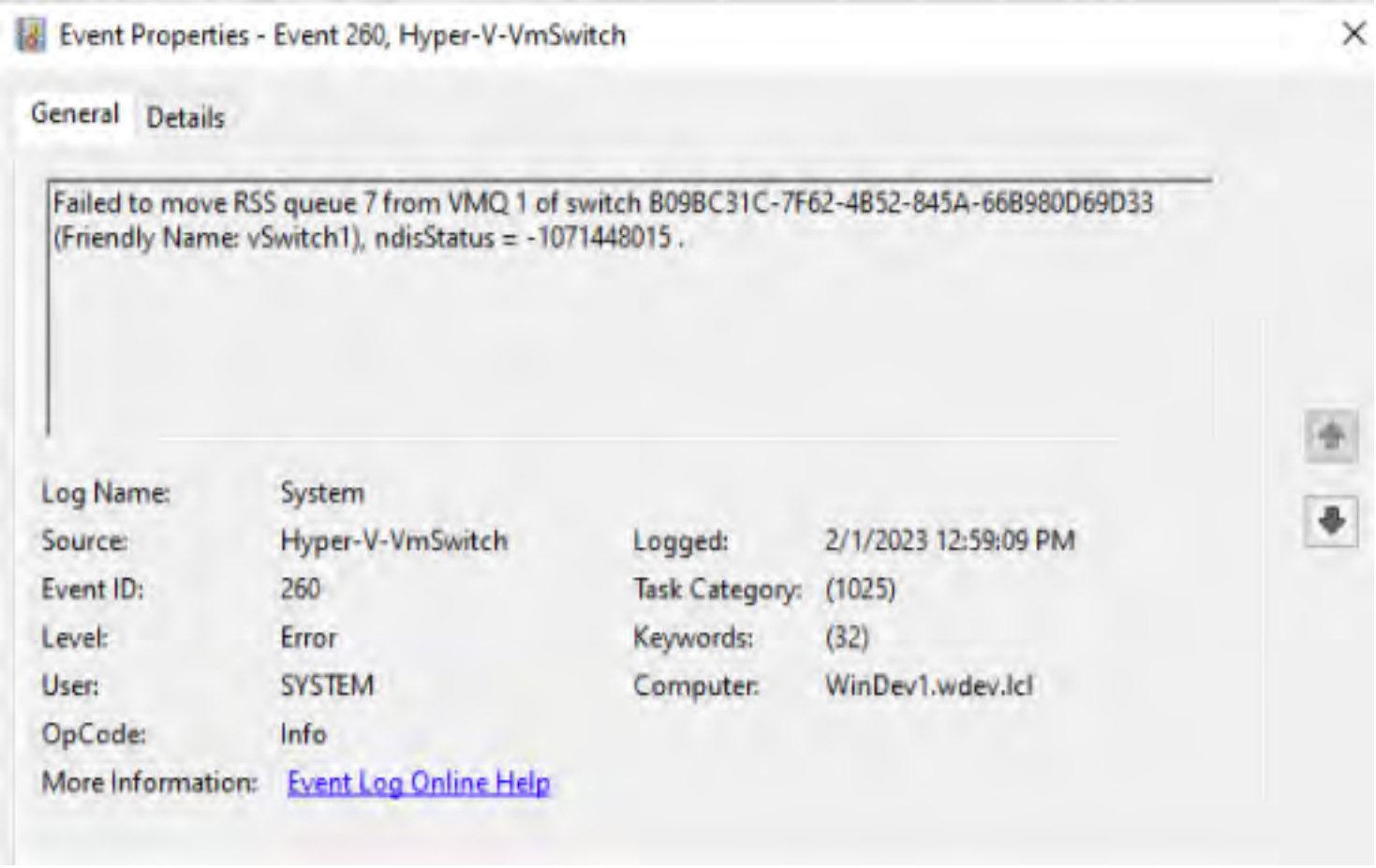
Windows Server Behavior

Event Properties - Event 260, Hyper-V-VmSwitch.

When using NXE for a Hyper-V vSwitch, Error 260 may be logged in the Event Viewer. This is the result of an RSSv2 indirection table update request from Windows that is incompatible with the current operational configuration. The driver

has reported this to the operating system with the appropriate status code. Hyper-V logs this as an error despite there being no functional issue.

The following figure shows an example of this error.



Event Properties - Event 280 - Hyper-V

When using NXE for a Hyper-V vSwitch, Error 280 may be logged in the Event Viewer. This warning is the result of the Number of QPairs being limited to 8 for non-default vPorts.

Updating the Firmware on Ethernet Network Adapters

Provides instructions for updating the firmware on Ethernet network adapters manually or using automated tools.

The NICCLI utility requires that the network driver is loaded and running to communicate with the adapter being upgraded. If the NICCLI utility does not recognize the adapter being upgraded, the network driver running on the system is too old and does not support that adapter. In this case, update the network driver by using the driver provided on the Broadcom website before using the NICCLI utility to update the adapter firmware.

Note: When upgrading the firmware on Linux, you must be logged in as root or use sudo when using NICCLI.

The controller firmware can be updated using one of the following methods.

- Updating the Firmware Online (Windows/Linux) – The tool automatically downloads the firmware from the Broadcom website.
- Updating the Firmware with the Automated Installer for Linux – The installer script updates the firmware and the drivers.
- Updating the Firmware on Linux/ESX Manually – Manually updates using the NICCLI utility.
- Updating the Firmware on Windows – Manually updates using the NICCLI utility on Windows.
- Verifying the Firmware – Verifies the firmware revision installed on the controller.

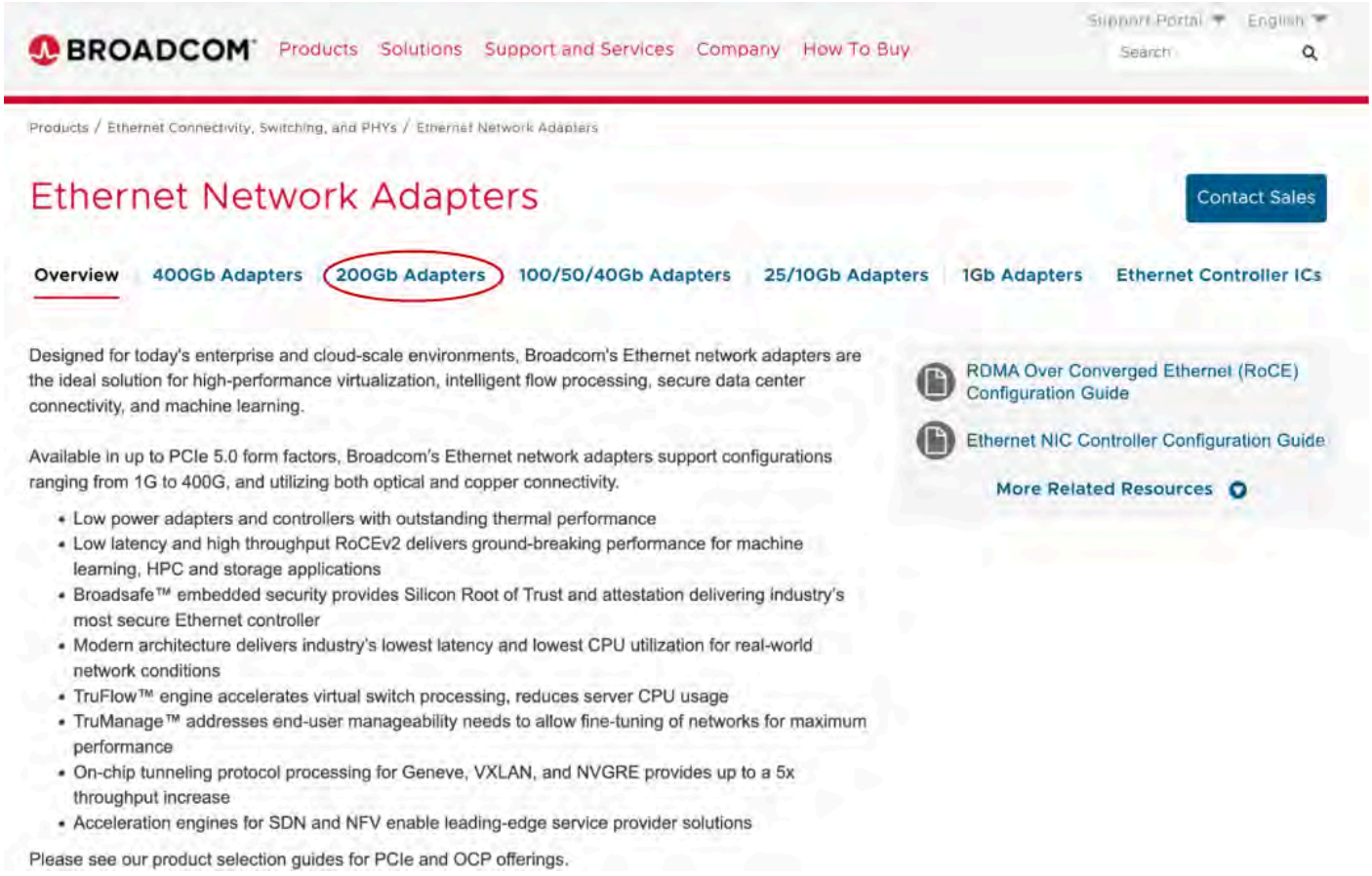
Updating the Firmware Online (Windows/Linux)

Provides instructions for updating the Ethernet network adapter firmware online (Windows/Linux).

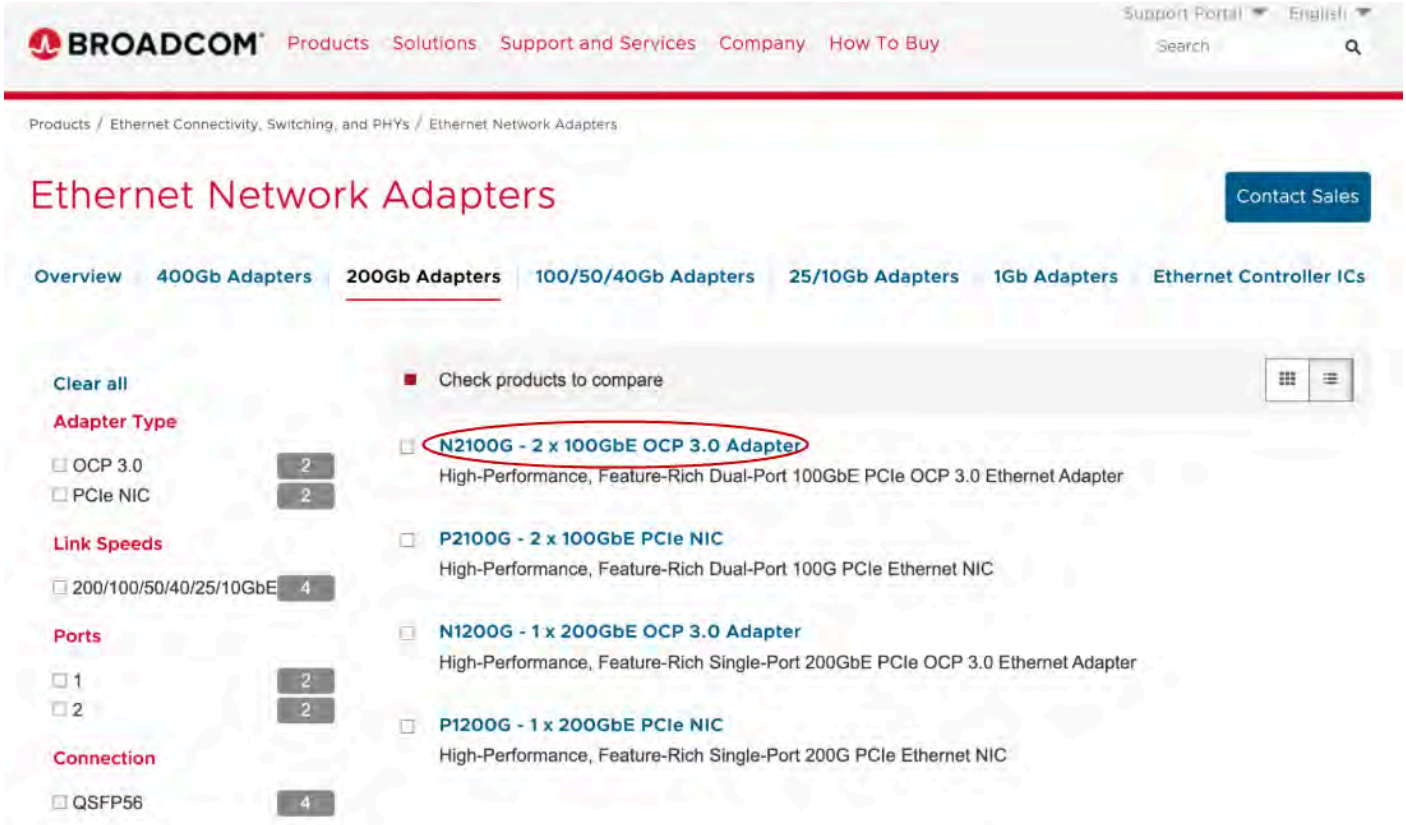
To update the firmware online, download the niccli tool used to upgrade the firmware from the Broadcom public website. The tool, when run on a server with Internet access, downloads the appropriate firmware package from the Internet and installs the package.

Use the following steps to update the firmware online:

1. Using a browser, access the following website: https://www.broadcom.com/products/ethernet-connectivity/network-adapters

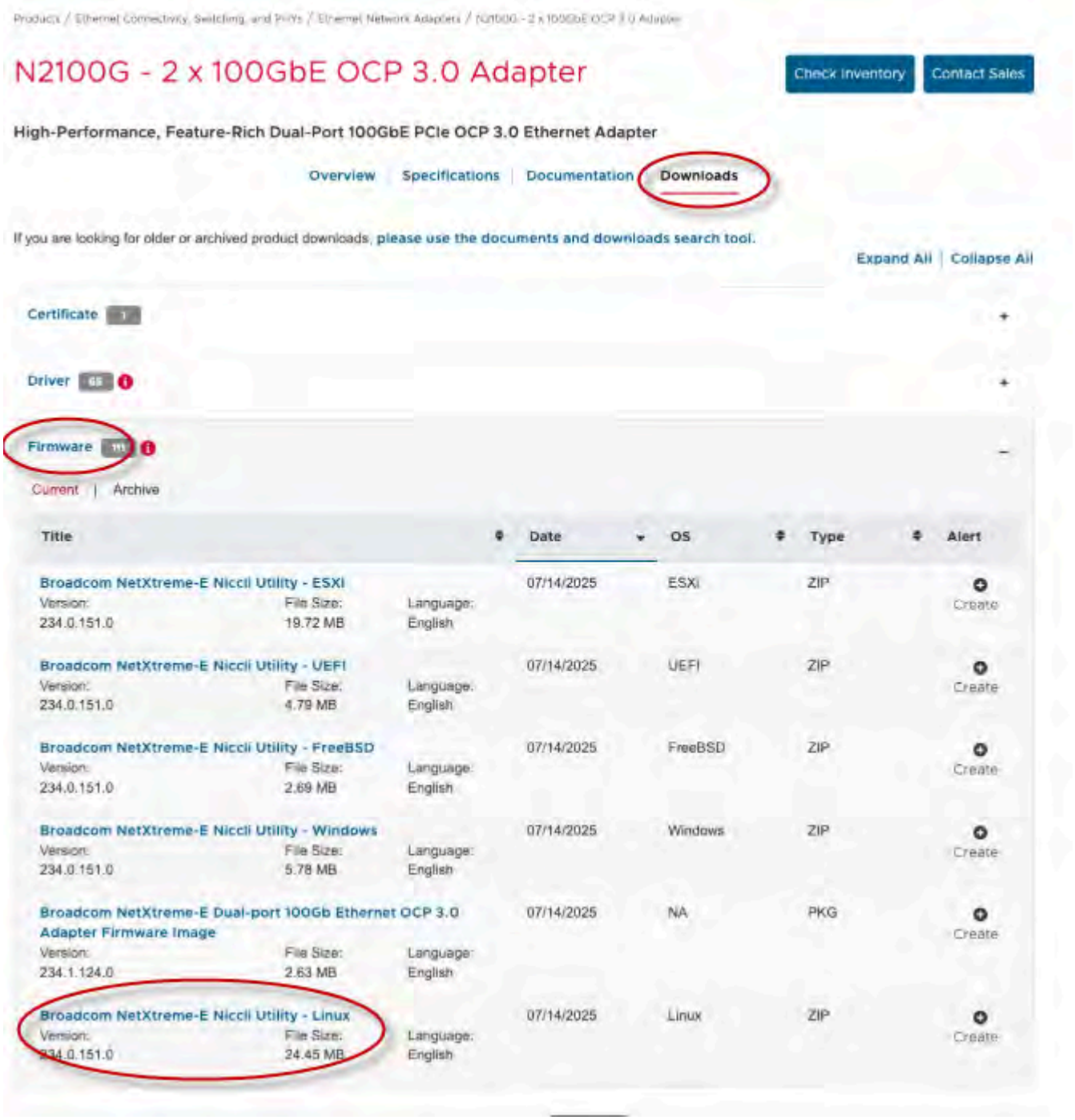


2. Select the adapter according to your port speed.



3. Select your specific adapter. The N2100G is selected in the previous figure as an example.
4. Click **Downloads** followed by **Firmware**.

5. Select **Broadcom NetXtreme-E NICCLI Utility - Linux (for Linux OS)** and download the upgrade tool.



6. Run the tool with NICCLI using the following instructions.
   a. `niccli -i 1 fw --update --online`

```
--------------------------------------------------------------------------------
NIC CLI v234.0.151.0 - Broadcom Inc. (c) 2025 (Bld-106.52.39.138.16.0)
--------------------------------------------------------------------------------

WARNING : Don't perform power cycle or reboot the system while firmware update
```

```
is in progress as the device may become inoperable.

Active Package Version - 235.1.101.0 : Package Version on NVM - 235.1.101.0

NetXtreme-E Controller at PCI Domain 0000:9c:00:0
Device 0000:9c:00:0 : Installing package file BCM14e41751.pkg
Device 0000:9c:00:0 : will be updated to package version 234.1.124.0

Do you want to continue (y/n)?y
Firmware Update is in progress. Please wait ...
##############################


Firmware update is completed.
A system reboot or device reset is needed for the firmware update to take effect.
```

Note: Reboot the system after the adapter has been upgraded with the new firmware package for the new firmware to take effect.

Updating the Firmware with the Automated Installer for Linux

Provides instructions for updating the firmware on Ethernet network adapters using the Linux automated installer.

This section is based on the `Broadcom NetXtreme-E Linux Installer` available from the Broadcom Ethernet Network Adapters site under the **Downloads** > **Driver** section located on each device page. Download and extract the driver to a directory referenced as `$REL_DIR` and use the instructions in Installing the L2 and RoCE Drivers Using the Automated Installer.

Note: The automated installer automatically upgrades the driver and the firmware. To upgrade only the firmware, see Manually Updating the Adapter Firmware on Linux/ESX.

Updating the Firmware Manually on Linux/ESX

Provides instructions for manually updating the firmware on Ethernet network adapters for Linux/ESX using NICCLI.

To update the adapter firmware on Linux/ESX using niccli, see the following sections:

Note: For the ESXi 8.x and 9.x, when the signed niccli package is used, all niccli commands must have `esxcli` added to the beginning of each command. For example, esxcli niccli --list_devices.

Updating the Firmware with NICCLI

Note: To install NICCLI, see Installing the NICCLI Configuration Utility.

1. List the adapters in the system using the following command:

   ```
   niccli --list_devices
   ```

   This command shows the device interface name that identifies the adapter in subsequent commands.

   ```
   1 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
   Device Interface Name : enp10s0np0
   MAC Address : 6C:92:CF:63:40:30
   PCI Address : 0000:0A:00.0

   2 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#1)
   Device Interface Name : eno12399np0
   MAC Address : 14:23:F2:32:46:80
   ```

```
PCI Address : 0000:1F:00.0

3 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#2)
Device Interface Name : eno12409np1
MAC Address : 14:23:F2:32:46:81
PCI Address : 0000:1F:00.1
```

2. After selecting the device interface name, the adapter part number can be viewed using the following command:

   ```
   niccli -i <index> show -d
   ```

   This command shows the part number of the Broadcom Ethernet device for which firmware has to be downloaded and upgraded.

```
NIC State : Up
Device Type : THOR2
PCI Vendor ID : 0x14E4
PCI Device ID : 0x1760
PCI Revision ID : 0x11
PCI Subsys Vendor ID : 0x14E4
PCI Subsys Device ID : 0x9140
Device Interface Name : enp10s0np0
MAC Address : 6C:92:CF:63:40:30
Base MAC Address : 6C:92:CF:63:40:30
Serial Number : P140024260003KFG
Part Number : BCM957608-P1400GDF00
PCI Address : 0000:0a:00.0
Chip Number : BCM57608
Chip Name : THOR2
Description : Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC
Firmware Name : PRIMATE_FW
Firmware Version : 233.0.150.0
RoCE Firmware Version : 233.0.150.0
HWRM Interface Spec : 1.10.3
Kong mailbox channel : Not Applicable
Active Package Version : 233.1.134.0
Package Version on NVM : 233.1.134.0
Active NVM config version : 0.0.15
NVM config version : 0.0.15
Reboot Required : No
Firmware Reset Counter : 0
Error Recovery Counter : 0
Crash Dump Timestamp : Not Available
Secure Boot : Enabled
Secure Firmware Update : Enabled
FW Image Status : Operational
Crash Dump Available in DDR : No
Device Temperature : 67 Celsius
PHY Temperature : Not Available
Optical Module Temperature : Not Available
Device Health : Good
```

3. Download the firmware package file from the corresponding device page **Downloads > Firmware** section.
4. The firmware is updated using the following command:

   ```
   niccli -i <index> fw --update -f <pkg file>
   WARNING : Don't perform power cycle or reboot the system while firmware update
   ```

```
is in progress as the device may become inoperable.

Active Package Version - 233.1.134.0 : Package Version on NVM - 233.1.134.0

NetXtreme-E Controller at PCI Domain 0000:0a:00:0
Device 0000:0a:00:0 : Installing package file ./BCM957608-P1400GDF00.pkg
Device 0000:0a:00:0 : will be updated to package version 234.1.124.0

Do you want to continue (y/n)?y
Firmware Update is in progress. Please wait ...
################################################


Firmware update is completed.
A system reboot is needed for firmware update to take effect.
FW package update SUCCESS!
```

5. Reboot the system after the adapter has been upgraded with the new firmware package for the new firmware to take effect.

Updating the Firmware on Windows

Provides instructions for updating the firmware on Ethernet network adapters for Windows.

Note: During a hot firmware upgrade or error recovery, there might be a period of time when the firmware cannot respond to a command sent from the drivers while the firmware performs a reset. The driver logs this as an error in the event log. During normal operation, the firmware continues the reset and returns to normal operation. As long as the firmware hot upgrade/error recovery completes successfully, this error can be ignored. Success is verified by observing that the *Firmware became responsive* and *Firmware reset sequence completed* events in the event log.

Use the following steps to update the adapter firmware on Windows:

Updating the Firmware with NICCLI

1. List the adapters in the system using the following commands:

```
niccli --list
niccli --list_devices
```

   This command shows the device interface name that identifies the adapter in subsequent commands.

```
1 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
Device Interface Name : enp10s0np0
MAC Address : 6C:92:CF:63:40:30
PCI Address : 0000:0A:00.0

2 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#1)
Device Interface Name : eno12399np0
MAC Address : 14:23:F2:32:46:80
PCI Address : 0000:1F:00.0

3 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#2)
Device Interface Name : eno12409np1
MAC Address : 14:23:F2:32:46:81
PCI Address : 0000:1F:00.1
```

2. After selecting the device interface name, the adapter part number can be viewed using the following command:

   ```
   niccli -i <index> show -d
   ```

   This command shows the part number of the Broadcom Ethernet device for which firmware has to be downloaded and upgraded.

   ```
   NIC State : Up
   Device Type : THOR2
   PCI Vendor ID : 0x14E4
   PCI Device ID : 0x1760
   PCI Revision ID : 0x11
   PCI Subsys Vendor ID : 0x14E4
   PCI Subsys Device ID : 0x9140
   Device Interface Name : enp10s0np0
   MAC Address : 6C:92:CF:63:40:30
   Base MAC Address : 6C:92:CF:63:40:30
   Serial Number : P140024260003KFG
   Part Number : BCM957608-P1400GDF01
   PCI Address : 0000:0a:00.0
   Chip Number : BCM57608
   Chip Name : THOR2
   Description : Broadcom BCM57608 1x400G QSFP-DD PCIe
   ```

3. Download the firmware package file from the corresponding device page **Downloads > Firmware** section.
4. The firmware is updated using the following command:

   ```
   niccli -i <index> fw --update -f <pkg file>
   WARNING : Don't perform power cycle or reboot the system while firmware update
   is in progress as the device may become inoperable.

   Active Package Version - 233.1.134.0 : Package Version on NVM - 233.1.134.0

   NetXtreme-E Controller at PCI Domain 0000:0a:00:0
   Device 0000:0a:00:0 : Installing package file ./BCM957608-P1400GDF00.pkg
   Device 0000:0a:00:0 : will be updated to package version 233.1.134.0

   Do you want to continue (y/n)?y
   Firmware Update is in progress. Please wait ...
   ################################################


   Firmware update is completed.
   A system reboot is needed for firmware update to take effect.
   FW package update SUCCESS!
   ```

5. Reboot the system after the adapter has been upgraded with the new firmware package for the new firmware to take effect.

Verifying the Firmware

Provides instructions for verifying the firmware on Ethernet network adapters.

After upgrading the firmware and rebooting, use the following commands to verify the installed firmware version and status:

```
niccli --list_devices
niccli --list
```

```
niccli -i 1 show --pkg_ver
niccli -i 1 show -d
niccli -i 1 nvm --list
```

niccli --listdev Example

```
brcm@dhcp-10-136-14-159 ~]$ sudo niccli --list_devices

1 ) Broadcom BCM57608 1x400G QSFP-DD PCIe Ethernet NIC (Adp#1 Port#1)
Device Interface Name : enp10s0np0
MAC Address : 6C:92:CF:63:40:30
PCI Address : 0000:0A:00.0

2 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#1)
Device Interface Name : eno12399np0
MAC Address : 14:23:F2:32:46:80
PCI Address : 0000:1F:00.0

3 ) Broadcom Adv. Dual 25Gb Ethernet (Adp#2 Port#2)
Device Interface Name : eno12409np1
MAC Address : 14:23:F2:32:46:81
PCI Address : 0000:1F:00.1
```

niccli --list Example

```
[brcm@dhcp-10-136-14-159 ~]$ sudo niccli --list

BoardId(Rev) MAC Address FwVersion PCIAddr Type Mode
1) BCM57608(B1) 6C:92:CF:63:40:30 233.0.150.0 0000:0A:00.0 NIC PCI
2) BCM57414(B1) 14:23:F2:32:46:80 223.0.183.0 0000:1F:00.0 NIC PCI
3) BCM57414(B1) 14:23:F2:32:46:81 223.0.183.0 0000:1F:00.1 NIC PCI
```

niccli -i 1 pkgver Example

```
[brcm@dhcp-10-136-14-159 ~]$ niccli -i 1 show --pkg_ver

Package Information :
Active Package Version : 233.1.134.0
Package Version on NVM : 232.1.132.8
Primary SBI Version : 232.0.7.0
Secondary SBI Version : 232.0.7.0
Primary SRT Version : 232.0.155.8
Secondary SRT Version : 233.0.150.0
Primary CRT Version : 232.0.155.8
Secondary CRT Version : 233.0.150.0
```

niccli -i 1 show Example

```
[brcm@dhcp-10-136-14-159 ~]$ niccli -i 1 show -d

NIC State : Up
Device Type : THOR2
PCI Vendor ID : 0x14E4
PCI Device ID : 0x1760
```

```
PCI Revision ID : 0x11
PCI Subsys Vendor ID : 0x14E4
PCI Subsys Device ID : 0x9140
Device Interface Name : enp10s0np0
MAC Address : 6C:92:CF:63:40:30
Base MAC Address : 6C:92:CF:63:40:30
Serial Number : P140024260003KFG
Part Number : BCM957608-P1400GDF01
PCI Address : 0000:0a:00.0
Chip Number : BCM57608
Chip Name : THOR2
Description : Broadcom BCM57608 1x400G QSFP-DD PCIe
```

nvm list Example

```
[brcm@dhcp-10-136-14-159 ~]$ niccli -i 1 nvm --list

Device Interface Name: enp10s0np0

item type ord.ext data/length attr version
1 MBA 1.0 348448/352256 0010 233.0.146.0
2 SRTImage 1.0 538208/540672 0000 233.0.150.0
3 SBIImage 0.0 449168/524288 0000 232.0.7.0
4 CrashDmpData 0.0 1048576/1048576 0001
5 MBA 0.0 358688/360448 0010 232.0.155.4
6 VPD 0.0 324/4096 0000
7 pkgLog 0.0 1292/4096 0000 232.1.132.8
8 systemCfg 0.0 36864/36864 0001
9 manufacturing 0.0 36864/36864 0001
10 CertChain 0.0 4144/8192 0000
11 SBIImage 1.0 449168/524288 0000 232.0.7.0
12 CrashDmpData 1.0 1048576/1048576 0001
13 factoryCfg 2.0 36864/36864 0001
14 CRTImage 1.0 2112080/2113536 0000 233.0.150.0
15 systemCfg 2.0 36864/36864 0001
16 CfgCRCs 0.0 8192/8192 0001
17 factoryCfg 0.0 36864/36864 0001
18 CRTImage 0.0 2079248/2113536 0000 232.0.155.8
19 SRTImage 0.0 517424/540672 0000 232.0.155.8
20 update 0.0 3443136/3489792 0001
```

Configuring Ethernet Network Adapters

Provides configuration information for various functions on Broadcom Ethernet network adapters ensuring optimum performance.

This section provides the following information on configuring the Ethernet network adapters.

- Tunneling Configuration Examples
- Link Aggregation on Ethernet Network Adapters
- Ethernet Physical Link Control Policy
- Configuring Auto-Negotiation on Ethernet Network Adapters
- Configuring Port Breakout on BCM95750X/BCM957608 Ethernet Adapters
- Configuring 200G Link Speeds on 2 x 100G BCM957508 Ethernet Network Adapters
- Configuring 200G Link Speeds on 2 x 100G BCM957608 Ethernet Network Adapters
- Configuring 200G and 400G Link Speeds on 2 x 200G BCM957608 Ethernet Network Adapters
- Configuring PXE Boot on Ethernet NIC Controllers
- Configuring SR-IOV and Use Case Examples
- Configuring RDMA SR-IOV for Ethernet NIC Controllers
- Configuring NPAR and Use Case Example
- Configuring DPDK Tunings
- Configuring DCBX – Data Center Bridging
- NICCLI Configuration Utility
- Configuring Peer Memory Direct with BCM5750X/BCM57608 Network Adapters
- Configuring RDMA over Converged Ethernet (RoCE)
- Precision Time Protocol

Tunneling Examples for Ethernet Network Adapters

Provides configuration examples for tunneling on Ethernet Network Adapters.

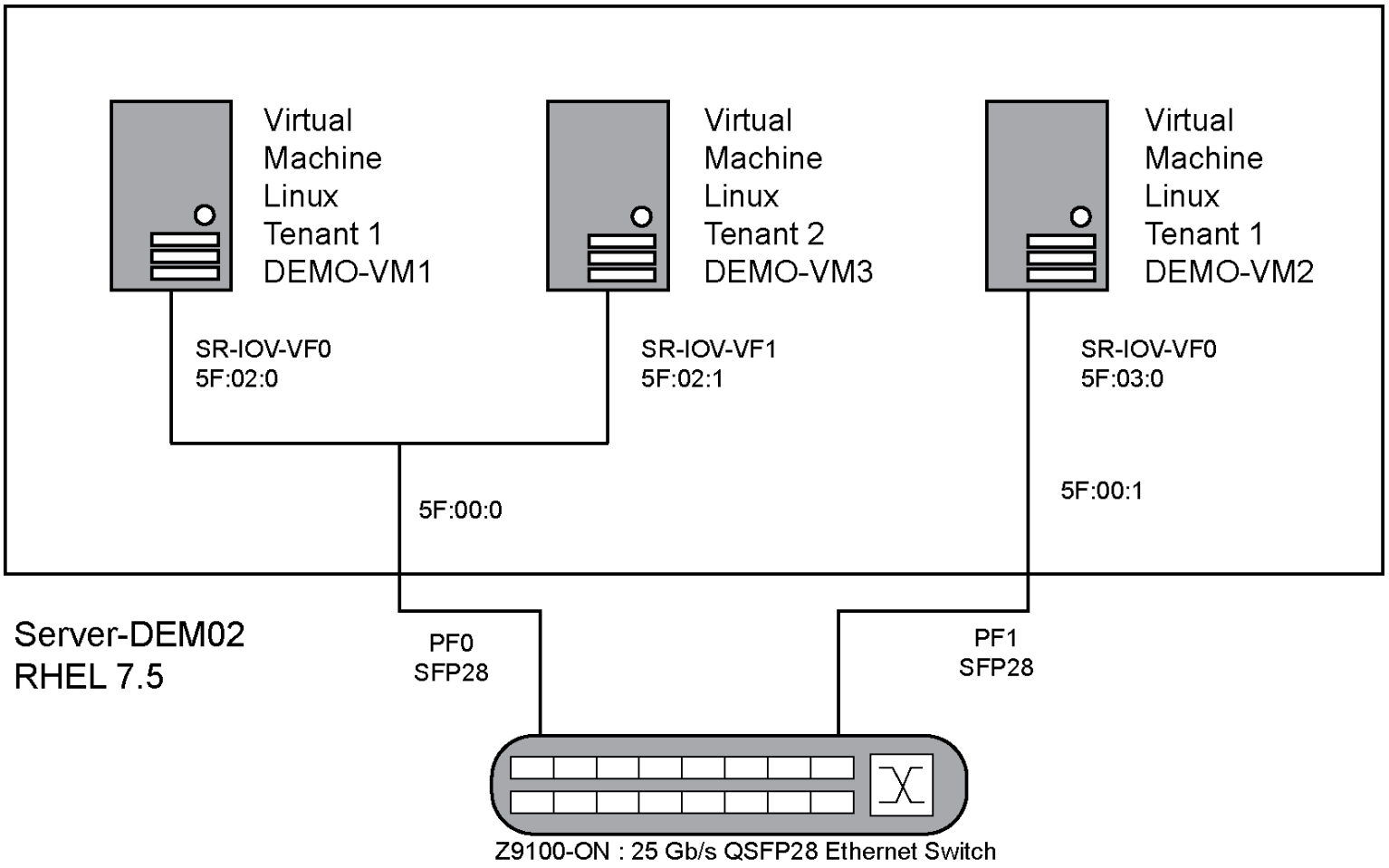
Broadcom BCM5741X and BCM575XX devices support VXLAN, GRE, and IP-in-IP tunneling offloads. This section provides the following tunneling configuration examples:

- Network Diagram
- VEB and VEPA Modes
- VLAN Double Tagging
- GRE Tunnelling
- IP-in-IP Tunnelling
- VXLAN – Configuration and Use Case Examples

Network Diagram

The test network shown in the following figure uses one Linux server with one two-port Ethernet adapter. SR-IOV is enabled in the adapter and two VFs are instantiated on the first port. A VF from the second port is exposed to the third VM (see SR-IOV – Configuration and Use Case Examples for information on SR-IOV bring up).

Figure 10: Network Diagram



VEB and VEPA Modes

Virtual Ethernet Bridging (VEB) mode generates an internal bridge within the NIC for VM-to-VM communication. The Ethernet frames are traversed through the internal bridge. Virtual Ethernet Port Aggregator (VEPA) mode transports the frames to the external switch. The switch handles the frame transport between the ports. VEB and VEPA can be configured through the UEFI HII.

VLAN Double Tagging

A VLAN can be configured at the PF level (see Device Configuration Menu) and another VLAN can be configured at the VF level inside the VM. After the VF is exposed inside the VM, the Linux L2 driver can be installed to activate the interface. Use the following commands to enable the VLAN inside the VM:

```
modprobe 8021q;
ip link add link <IntName> name <IntName.vlan_num> type vlan id <vlan_num>
ip link set <IntName.vlan_num> up
ip addr add <IPAddr>/mask broadcast <Gateway Addr> dev <IntName.vlan_num>
```

GRE Tunneling

An IP GRE is an IP inside an IP tunnel that can carry private network traffic between two heterogeneous networks. On the VM, use the following commands:

```
modprobe ip_gre;
ip tunnel add gre45 mode gre local <public IP> remote <Private IP>
ip link set dev gre45 up;
ip addr add <private IP>/mask broadcast <broadcast ip> dev gre45
```

In this example, gre45 is the interface name.

IP-in-IP Tunnelling

Similar to GRE, IP-in-IP is another encapsulation that carries a private IP onto a public IP. Use the following commands:

```
modprobe ipip
ip tunnel add ipip45 mode ipip remote <Peer IP> local <private ip> ttl 255 dev <VM Interface Name>
ip addr add dev ipip45 <IP addr> peer <Peer IP> /mask
ip link set dev ipip45 up
```

In this example, ipip45 is the name of the newly created tunnel device.

VXLAN – Configuration and Use Case Examples

VXLAN encapsulation permits multiple virtual machines or containers residing on one server to be isolated from each other in virtual tunnels by encapsulating traffic with VXLAN headers. Broadcom Ethernet NIC controllers accelerate this encapsulation and de-encapsulation in hardware.

This example discusses basic VXLAN connectivity between two Linux servers. Each server has one physical NIC enabled with the outer IP address set to 1.1.1.4 and 1.1.1.2, respectively.

A VXLAN interface with ID 10 is created with multicast group 239.0.0.10 and is associated with physical network port eth1 on each server.

An IP address for the host is created on each server and associated with that VXLAN interface. After the VXLAN interface is brought up, the VM present in system 1 can communicate with the VM present in system 2 using the VXLAN interfaces. The VLXAN format is shown in the following table.

Table 23: VXLAN Frame Format

MAC header	Outer IP header with proto = UDP	UDP header with Destination port= VXLAN	VXLAN header (Flags, VNI)	Original L2 Frame	FCS

The following table provides VXLAN command and configuration examples.

Table 24: VXLAN Command and Configuration Examples

System 1	System 2
ifconfig eth1 1.1.1.4/24	ifconfig eth1 1.1.1.2/24
ip link add vxlan10 type vxlan id 10 group 239.0.0.10 dev eth1 dstport 4789	ip link add vxlan10 type vxlan id 10 group 239.0.0.10 dev eth1 dstport 4789
ifconfig vxlan10 192.168.1.5 mtu 1450	ifconfig vxlan10 192.168.1.10 mtu 1450
ip --d link show vxlan10	—

System 1	System 2
ping 192.168.1.10	—

Link Aggregation on Ethernet Network Adapters

Provides information on link aggregation in Windows and Linux operating systems.

Windows

Broadcom Ethernet network adapters can aggregate network links using the Microsoft teaming feature. For more information on the adapter teaming functionality, refer to the Microsoft public documentation on Microsoft.com.

Linux

The Linux bonding module is used for link aggregation under Linux. For additional documentation on Linux bonding, refer to https://www.kernel.org/doc/Documentation/networking/bonding.txt.

Configuring a RoCE Link Aggregation Group

Provides information on configuring RoCE link aggregation group.

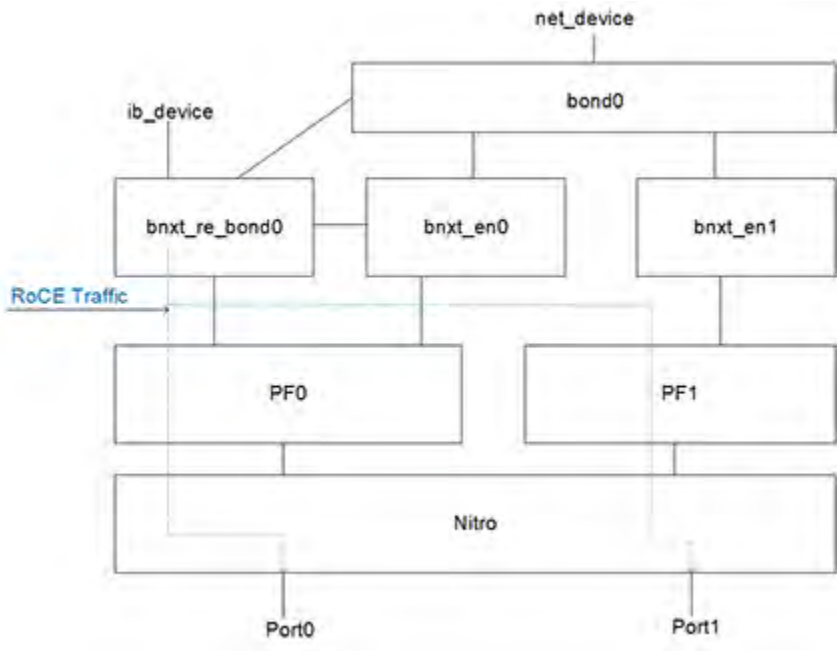
Link aggregation is a common technique used to provide additional aggregate bandwidth and high availability for logical interfaces that aggregate multiple physical interfaces. Additional aggregate bandwidth can be achieved by balancing the traffic load across multiple physical interfaces. High availability can be achieved by reconfiguring the loads across the active links when one of the physical links fails.

The bond driver provides the following multiple modes that affect how the bond interface operates:

- **Round-robin or mode 0** - The packets are transmitted in sequential order from the first available interface through the last. This mode provides load balancing and fault tolerance.
- **Active-backup or mode 1** - All traffic uses the active link. If the active link fails, all traffic is redirected to the backup link.
- **Balance-xor or mode 2** - A hash algorithm is applied to the traffic to determine which link to use for a given flow. If any of the links fail, then the hash table is updated so that traffic only resolves to active links.
- **802.3ad (LACP) or mode 4** - This mode is similar to the balance-xor mode, however, it uses LACP to manage when interfaces are added to or removed from the LAG.

These concepts of link aggregation can also be applied to RoCE.

Figure 11: RoCE Link Aggregation Group



The current solution allows a link aggregation only if all of the following conditions are met:

- The netdev associated with each RDMA interface is a part of an upper level device.
- The two netdev interfaces are part of the same bond device.
- Two netdevs on the same physical device are added to the bond.
- The link aggregate cannot span separate physical devices.
- The bond interface has exactly two non-NPAR physical interfaces.
- The bond mode is one of the following modes:
  - round-robin (mode 0)
  - active-backup (mode 1)
  - xor (mode 2)
  - 802.3ad (mode 4)

RoCE LAG is dependent on having the associated Ethernet interfaces bonded by standard Linux mechanism.

When in RoCE LAG mode, instead of having a RoCE device per physical port (for example `bnxt_re_0` and `bnxt_re_1`), only one RoCE device is present for both ports with **bond** appended to its name (for example `bnxt_re_bond0`). This device provides an aggregation of both RoCE ports, just as the bond interface provides an aggregation of both Ethernet interfaces.

Hardware Link Aggregation

The BCM57608 device includes hardware-based Link Aggregation (LAG) feature that supports LAG on VFs. Hardware LAG does not use the `xmit_hash_policy` specified by the Linux bonding driver. Hardware LAG enablement is controlled via NVM config option `enable_hw_lag` . This NVM option is a global configuration and can be configured by the NICCLI tool. After setting this option, a warm reboot is required for the option to take effect. To set and get the NVM config option using NICCLI are:

```
niccli -i <index> nvm --getoption enable_hw_lag
niccli -i <index> nvm --setoption enable_hw_lag --value <value>
0 - HW LAG disabled
```

```
1 - HW LAG enabled
Example below:
niccli -i 1 nvm --setoption enable_hw_lag --value 1
enable_hw_lag is set successfully
Please reboot the system to apply the configuration

niccli -i 1 nvm --getoption enable_hw_lag

enable_hw_lag = True
```

There are two modes of hardware LAG supported in the BCM57608 which can be changed via NVM config option `hw_lag_perf_mode` . This NVM option is a global configuration and can be configured by the NICCLI tool. After setting this option, a warm reboot is required for the option to take effect

Hardware LAG Modes:

- **queue_affinity_mode** - This is the default mode. For the `queue_affinity` mode, when mode 2 or 4 is specified, a round-robin (RR) distribution algorithm is followed. The current RR algorithm achieves load balancing on a per DPI (application) basis.
- **Hash_mode** - In `Lag_hash` mode, the hardware hash uses a Lookup3 LAG hash calculation over the following inner header tuples:
  - Non-IP packets: DMAC/SMAC/Ethertype
  - IP only packets: SIP/DIP/Protocol
  - UDP/TCP packets: SIP/DIP/Protocol/L4 source port/L4 destination port

To set and get the NVM config option using NICCLI:

```
niccli -i <index> nvm --getoption hw_lag_perf_mode
niccli -i <index> nvm --setoption hw_lag_perf_mode --value <value>
0 - queue_affinity_mode
1 - hash_mode
Example below:
niccli -i 1 nvm --setoption hw_lag_perf_mode --value 0

hw_lag_perf_mode is set successfully
Please reboot the system to apply the configuration

niccli -i 1 nvm --getoption hw_lag_perf_mode

hw_lag_perf_mode = 0x0 (0)
```

Configuring RoCE LAG

Both the BCM5750X and BCM57608 devices support RoCE LAG, however, only the BCM57608 device supports hardware LAG. Step 2 in the following procedure is only applicable to BCM57608 devices.

1. Ensure that the `bnxt_en` and `bnxt_re` drivers are loaded before creating the bond.
2. Enable hardware LAG by configuring the NVM config option `enable_hw_lag` . See the previous hardware LAG section on how to configure the NVM option.
3. Ensure bonding is enabled on the server by running `modprobe bonding` . Follow the distribution OS manuals to create a `bond0` interface. The sample bond configuration using netplan in Ubuntu 22.04 OS is as follows. For bonding configuration in RHEL OS, see here.

```
network:
 version: 2
 renderer: NetworkManager
```

```
ethernets:
  enp33s0f0np0:
    mtu: 9000
  enp33s0f1np1:
    mtu: 9000
bonds:
  bond0:
    mtu: 9000
    addresses: [192.10.10.10/24]
    interfaces:
      - enp33s0f0np0
      - enp33s0f1np1
    parameters:
      mode: 802.3ad
      mii-monitor-interval: 100
      transmit-hash-policy: layer2+3
```

In the previous example, the bond configuration is 802.3ad (mode 4). Similar to configuring other bonding modes, change the mode parameter to either active-backup (mode 1) or balance-xor (mode 2).

4. Run `netplan apply` for the bond configuration to take effect.
5. `Bnxt_re_bond0` is created and can be confirmed using `ibv_devinfo` command.

```
~# ibv_devinfo -d bnxt_re_bond0
hca_id: bnxt_re_bond0
       transport:                      InfiniBand (0)
       fw_ver:                         231.2.63.0
       node_guid:                      0415:f4ff:fe3e:dee0
       sys_image_guid:                 0415:f4ff:fe3e:dee0
       vendor_id:                      0x14e4
       vendor_part_id:                 5984
       hw_ver:                         0x9120
       phys_port_cnt:                  1
               port:   1
                       state:                  PORT_ACTIVE (4)
                       max_mtu:                4096 (5)
                       active_mtu:             4096 (5)
                       sm_lid:                 0
                       port_lid:               0
                       port_lmc:               0x00
                       link_layer:             Ethernet
```

6. Check the current status of `bond0`.

```
cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v6.8.0-49-generic

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2+3 (2)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0
```

```
802.3ad info
LACP active: on
LACP rate: slow
Min links: 0
Aggregator selection policy (ad_select): stable
System priority: 65535
System MAC address: 06:15:f4:3e:de:e0
Active Aggregator Info:
       Aggregator ID: 1
       Number of ports: 2
       Actor Key: 31
       Partner Key: 31
       Partner Mac Address: 9a:f6:c3:49:77:2c

Slave Interface: enp33s0f0np0
MII Status: up
Speed: 200000 Mbps
Duplex: full
Link Failure Count: 2
Permanent HW addr: d4:04:e6:7c:2b:d0
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 0
Partner Churned Count: 1
details actor lacp pdu:
   system priority: 65535
   system mac address: 06:15:f4:3e:de:e0
   port key: 31
   port priority: 255
   port number: 1
   port state: 61
details partner lacp pdu:
   system priority: 65535
   system mac address: 9a:f6:c3:49:77:2c
   oper key: 31
   port priority: 255
   port number: 1
   port state: 61

Slave Interface: enp33s0f1np1
MII Status: up
Speed: 200000 Mbps
Duplex: full
Link Failure Count: 2
Permanent HW addr: d4:04:e6:7c:2b:d1
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 1
Partner Churned Count: 1
```

```
details actor lacp pdu:
   system priority: 65535
   system mac address: 06:15:f4:3e:de:e0
   port key: 31
   port priority: 255
   port number: 2
   port state: 61
details partner lacp pdu:
   system priority: 65535
   system mac address: 9a:f6:c3:49:77:2c
   oper key: 31
   port priority: 255
   port number: 2
   port state: 61
```

Known Limitations

RoCE link aggregation groups have the following limitations:

1. A RoCE LAG is not supported on multi-host (MH) or multi-root (MR) platforms.
2. A RoCE bond is created only if there are two Ethernet functions added to the bond and the Ethernet devices are from the same physical adapter. Multiple adapters are not supported.
3. If NPAR is enabled, hardware LAG is not supported.
4. The BCM5750X does not support RoCE LAG on VFs.
5. The BCM57608 does not support RoCE LAG mode Round-Robin or mode 0.
6. Changing bond modes when a RoCE driver is in use can cause a system hang. Ensure that no reference to `bnxt_re` while changing the bond mode, which can be checked using the following command:
   ```
   lsmod | grep bnxt_re
   ```
7. Ensure that enough delay is provided (for example, 5-10 sec) when creating and destroying the bond. This is to avoid hang and call traces related to the `rtnl_lock` usage.

Ethernet Physical Link Control Policy

Provides information for physical link control policy on Ethernet network adapters.

In general, users are allowed to bring up/down the Ethernet Physical link on the host side via the network configuration utility, for example, ifconfig <intf> down/up, ifdown/ifup <intf>, ip link set <intf> down/up, and so forth.

Regarding Broadcom Ethernet network adapter physical link policy, there are some limitations controlling the Ethernet Physical link on the host side. Users can bring down the physical link if one of the following conditions is met.

1. The NVM configuration **NC-SI over RMII** disabled and it is determined by the adapter (for example, it is fixed for a particular adapter and cannot be changed by the user).
2. The NVM configuration **NC-SI over RMII** enabled and the management controller (BMC) issued the command `Disable Channel Command (0x04)` with ALD field enabled.
3. The NVM configuration `HOST PORT CONTROL` is enabled and it can be controlled by niccli utility. `niccli -i <index> nvm --getoption total_host_port_control --scope <port number>`

The following table shows the descriptions of those NVM options.

Table 25: NVM Options

NVM Configuration	Type	Description
NC-SI over RMII	Share	A binary CFG that enables or disables manageability features (for example, NC-SI, PLDM) over the sideband RMII interface. Enabling it allows the BMC to communicate with the NIC over RBT (RMII Based Transport). In general, LOM design or OCP3 adapters have this CFG enabled.
HOST Port Control	Port	When enabled, then the port and link configurations are completely controlled by the host(s) when one or more drivers are loaded on the port. When the port link is up, both host and Out-Of-Band (OOB) management traffic is supported on the port. A port link down command from the host OS or driver completely shuts down the PHY/link and ceases all traffic including the management traffic.

The following description is for the ALD field of NC-SI `Disable Channel Command (0x04)` command:

Figure 12: ALD Field of NC-SI

The 1-bit Allow Link Down (ALD) field can be used by the Management Controller to indicate that the specified channel will not be required to handle Pass-through traffic while disabled. The Network Controller is allowed to take down the external network physical link if no other functionality (for example, host OS or WoL [Wake-on-LAN]) is active.

Possible values for the 1-bit ALD field are as follows:

- `0b` = Keep link up for Pass-through management traffic
- `1b` = Allow link to be taken down

The following figure illustrates the packet format of the disable channel command:

Figure 13: Packet Format

	Bits			
Bytes	**31..24**	**23..16**	**15..08**	**07..00**
00..15	NC-SI Header			
16..19	Reserved			ALD
20..23	Checksum			
24..45	Pad			

Configuring Auto-Negotiation on Ethernet Network Adapters

Provides configuration information for auto-negotiation on Ethernet network adapters.

Note: When NPAR is enabled on BCM95741X devices, the port speed is preconfigured and cannot be changed by the driver.

Note: For 10GBASE-T network adapters, auto-negotiation must be enabled.

The Broadcom Ethernet NIC controller supports the following auto-negotiation features:

- Link speed auto-negotiation
- FEC auto-negotiation
- Pause/Flow Control auto-negotiation

Note:

- When using SFP+, and SFP28 connectors, use DAC cables that support auto-negotiation and ensure that the link partner port has been set to the matching auto-negotiation protocol. For example, if the local Broadcom port is set to IEEE 802.3by AN protocol, the link partner must support auto-negotiation and must be set to IEEE 802.3 by auto-negotiation protocol.
- When Media Auto Detect and IEEE 802.by + Consortium are enabled, the controller auto-detects the Ethernet cables and transceivers to start auto-negotiation with the link partner. FEC is negotiated during auto-negotiation and forced FEC does not take effect in this mode. The default setting enables IEEE 802.by + Consortium and Media Auto Detect.

The supported combination of link speed settings for two-port Ethernet NIC controllers is shown in the following table.

Table 26: Supported Link Speeds for the Dual Port BCM95741X and BCM95750X

Dual-Port	BCM95741X	BCM95750X
1G, 1G	Yes	No
1G, 10G	Yes	No
1G, 25G	Yes	No
10G, 10G	Yes	Yes
10G, 25G	No	Yes
25G, 25G	Yes	Yes
50G, 50G	N/A	Yes
100G, 100G	N/A	Yes

Table 27: Supported Link Speeds for the Quad Port BCM95750X

Quad-Port	BCM95750X
10G, 10G, 10G, 10G	Yes
10G, 10G, 10G, 25G	Yes
10G, 10G, 25G, 25G	Yes
10G, 25G, 25G, 25G	Yes
25G, 25G, 25G, 25G	Yes

- AN – Auto-negotiation.
- No AN – Forced speed.
- The BCM5750x controller supports independent port speeds – both NRZ and PAM4. Each port can be configured as forced or auto-negotiate to any speed supported by the device.

BCM95750X PAM-4 Configuration

- PAM-4 support is enabled with the following notes/restrictions:
  - PAM-4 requires FEC mode as RS544_1xN for 50G and 100G operation.
  - PAM-4 operation on DAC/Twinax cables requires link training to be enabled. This requires a switch that supports link training and PAM-4.

Note: A PAM-4 enabled switch must be used for PAM-4 signaling.

Note: When PAM-4 speeds are enabled on at least one port on a multi-port adapter, 10G and 40G speeds are disabled on all the ports.

Enabling PAM-4 Support

PAM-4 can be enabled on the required port from the UEFI HII or from the operating system.

From the UEFI HII Menu:

Enable PAM-4 support from the UEFI HII by setting **Autodetect Speed Exclude Mask** to **0** as shown in the following figure.

Figure 14: Enabling PAM-4 from the UEFI HII Menu

From the Operating System:

Use the following commands to read and set the `autodetect_speed_exclude_mask` to enable PAM-4 support.

NICCLI Commands:

- To read the current setting:

  ```
  niccli -i <index> nvm --getoption autodetect_speed_exclude_mask --scope <scope value> --value 0x0
  ```

  <scope value> is the port number.
- Enable PAM-4 support:

  ```
  niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope <scope value> --value 0x0
  ```

  <scope value> is the port number.

Command Examples
NICCLI Examples:

- Enable PAM-4 support on port 0:

```
niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope 0 --value 0x0
```

- Enable PAM-4 support on port 1:

```
niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope 1 --value 0x0
```

A system reboot is required for this change to take effect.

Disabling PAM-4 Speeds

To disable PAM-4 speeds, set `autodetect_speed_exclude_mask` to `0x1C0`.

To disable PAM-4 support, use the following command: `niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope <scope value> --value 0x1C0`

Configuring PAM-4 on Linux

- Currently, the Linux kernel and associated tools have the infrastructure to report link mode. From the reported link mode, the encoding or lanes that can be used for that link are derived. In a case where this is auto-negotiated, this works well. A device reporting a link mode of 100000baseCR2/Full uses a different number of lanes (and therefore encoding) than a device reporting 100000baseCR4/ Full. When auto-negotiation is not used, ample infrastructure does not exist to specify speed and encoding and differentiate between a card using NRZ versus PAM-4 encoding. Currently, ethtool supports setting the speed, but does not have support for setting the encoding or lanes used.

Configuring PAM-4 on VMware

PAM-4 operation on VMware requires that 50/100G PAM-4 is configured via the HII menu and also configured via ESXCLI commands. A reboot is required to ensure proper persistence of these parameters. The command is as follows:

```
esxcli network nic set -n <interface> -S <speed> -D full
```

Example:

```
esxcli network NIC set -n vmnic4 -S 200000 -D full
```

Configuring PAM-4 on Windows

For 200G speeds and above, the Broadcom Ethernet controllers use PAM-4 automatically. For speeds supported by both NRZ and PAM-4 (50G/100G), Windows defaults to NRZ. Configure your 50G/100G controller to use PAM-4 using the following steps:

1. From the **Windows Device Manager**, right-click on the Broadcom Ethernet Controller and select **Properties**.
2. Select the **Details** tab and then the **Driver Key** from the Property drop-down list.
3. Record the last 4 digits after the backslash.
4. Access the Windows registry editor (`regedit.exe`) and navigate to `HKEY_LOCAL_MACHINE\SYSTEM \CurrentControlSet\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}\xxxx` . The xxxx is the number recorded from the previous step.
5. Add a DWORD type value name `PreferredSignalingMode` and set the value to 1.
6. Save and exit the Windows registry editor.
7. Restart the driver (enable/disable).

The firmware sets the WMI field according to the link and can be used to check which signaling is currently being used.

```
Get-WmiObject -Namespace root\wmi BRCM_PhyConfig | Select-Object InstanceName,SignalMode
```

- SignalMode = 0 (NRZ)
- SignalMode = 1 (PAM-4)

Expected Link Speeds

The expected link speeds based on the local and link partner settings are shown in the following tables:

Table 28: Forced Speeds: Expected Link Speeds

Local Speed Settings	Link Partner Speed Settings						
	Forced 1G	**Forced 10G**	**Forced 25G**	**Forced 50G**	**Forced 100G**	**Forced 200G**	**AN Enabled**
Forced 1G	**1G***	No link	No link	No link	No link	No link	No link
Forced 10G	No link	**10G***	No link	No link	No link	No link	No link
Forced 25G	No link	No link	**25G**	No link	No link	No link	No link
Forced 50G	No link	No link	No link	**50G**	No link	No link	No link
Forced 100G	No link	No link	No link	No link	**100G**	No link	No link
Forced 200G	No link	No link	No link	No link	No link	**200G**	No link

* For 10GBASE-T network adapters, forced speeds are not supported. Auto-negotiation must be enabled. For SFP-based network adapters, Auto-negotiation is not supported. The link speed should either be configured for forced speeds or use Auto-negotiation + Media Auto-Detect.

Table 29: Auto-Negotiation: Expected Link Speeds

Link Speed Settings	Link Partner Speed Settings								
	AN Enabled 1 G	AN Enabled 10 G	AN Enabled 25G	AN Enabled1/10G	AN Enabled 1/25G	AN Enabled 10/25G	AN Enabled 1/10/25G	AN Enabled 10/25/50G	AN Enabled 10/25/50/100G
AN 1G	**1G***	No link	No link	1G	1G	No link	1G	No link	No link
AN 10G	No link	**10G***	No link	10G*	No link	10G*	10G*	10G*	10G*
AN 25G	No link	No link	**25G**	No link	25G	25G	25G	25G	25G
AN 1/10G	1G	10G*	No link	**10G***	1G	10G*	10G*	10G*	10G*
AN 1/25G	1G	No link	25G	1G	**25G**	25G	25G	25G	25G
AN 10/25G	No link	10G*	25G	10G*	25G	**25G**	25G	25G	25G
AN 1/10/25G	1G	10G*	25G	10G*	25G	25G	**25G**	25G	25G
AN 10/25/50G	1G	10G*	25G	10G*	25G	25G	25G	**50G**	50G

Link Speed Settings	Link Partner Speed Settings								
AN 10/25/50/100G	No link	10G*	25G	10G*	25G	25G	25G	50G	**100G**

* For 10GBASE-T network adapters, auto-negotiation must be enabled. Forced speeds are not supported. For SFP-based network adapters, Auto-negotiation is not supported. The link speed should either be configured for forced speeds or use Auto-negotiation + Media Auto-Detect.

To enable link speed auto-negotiation, the following options can be enabled in the system BIOS HII menu: **Main Configuration Menu** > **Device Configuration Menu**

Configuring Port Breakout on BCM95750X/BCM957608 Ethernet Adapters

Provides information on configuring port breakout on BCM95750X/BCM957608 Ethernet adapters.

Note: Before changing the **Port Operation Mode**, it is recommended that the NVM cfg file be reset to the factory default settings. Resetting the NVM cfg file can be done using the NICCLI command (see NICCLI Configuration Utility Usage and Commands).

Port breakout mode allows a high-speed port to connect to multiple low-speed ports. The following table provides a list of possible port breakout scenarios.

Table 30: Port Breakout Mode Scenarios

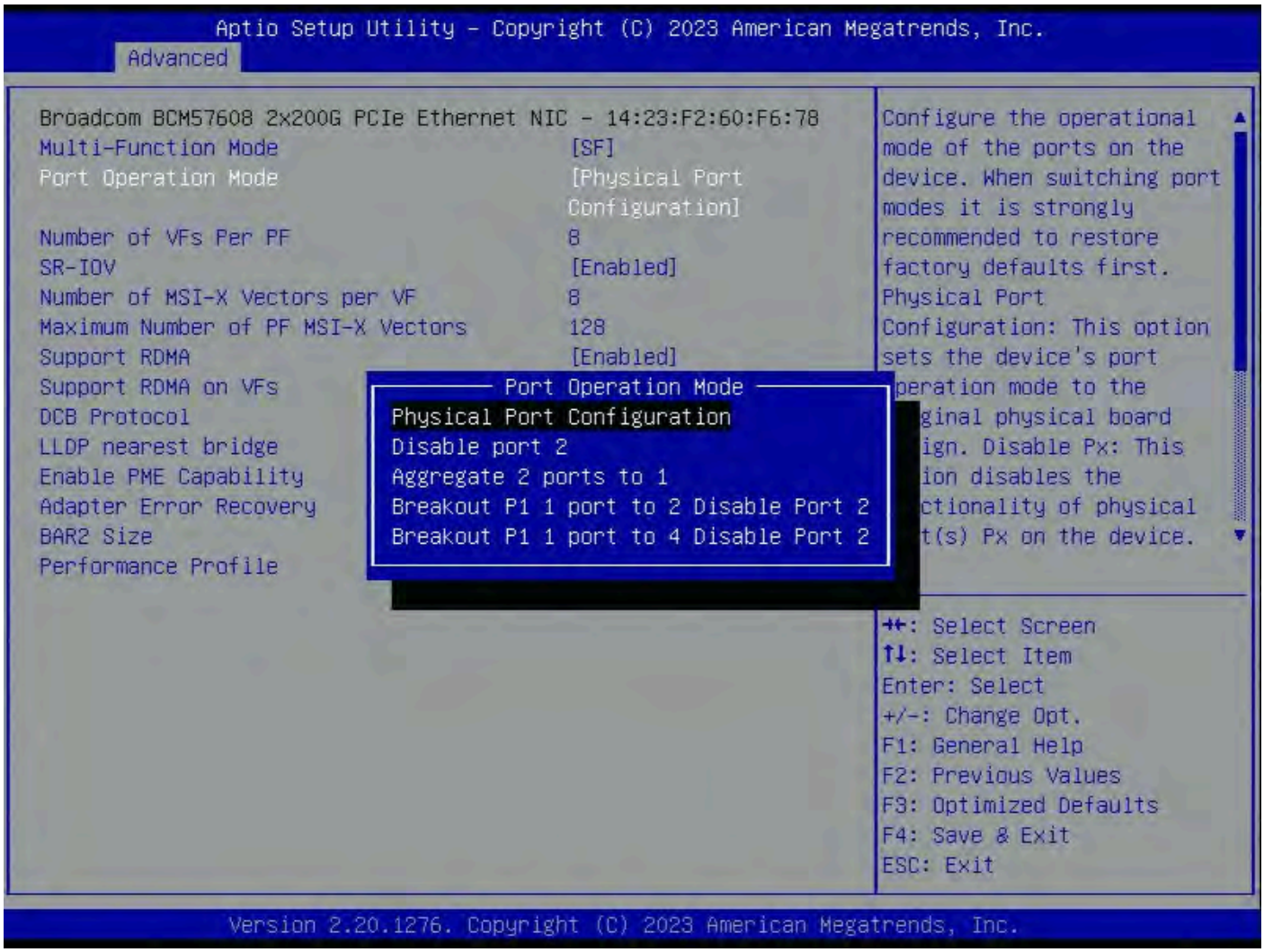
Broadcom Ethernet Adapters	Two Port Breakout	Four Port Breakout
BCM957504-N1100G (single-port, 100G Adapter)	2 x 50G	4 x 25G
BCM957508-P1200G (single-port, 200G Adapter)	2 x 100G	4 x 50G
BCM957508-N1200G (single-port, 200G Adapter)	2 x 100G	4 x 50G
BCM957608-P1400GD (single-port, 400G Adapter)	2 x 200G	4 x 100G
BCM957608-N1400GD (single-port, 400G Adapter)	2 x 200G	4 x 100G
BCM957508-P2100G (dual-port, 100G Adapter)	2 x 100G on first physical port, second port is disabled.	• 4 x 50G across two physical ports. • 4 x 50G on first physical port, second port is disabled.
BCM957508-N2100G (dual-port, 100G Adapter)	2 x 100G on first physical port, second port is disabled.	• 4 x 50G across two physical ports. • 4 x 50G on first physical port, second port is disabled.
BCM957608-P2200G (dual-port, 200G Adapter)	2 x 200G on first physical port, second port is disabled.	4 x 100G on the first port, second port is disabled.
BCM957608-N2200G (dual-port, 200G Adapter)	2 x 200G on first physical port, second port is disabled.	4 x 100G on the first port, second port is disabled.

Configuring Port Breakout Using the UEFI

Refer to the previous table to determine the appropriate breakout configuration for your Ethernet Adapter. To configure Port Breakout using the UEFI:

1. Set **Port Operation Mode** to either **Breakout 1 port to 2** or **Breakout 1 port to 4** as shown in the following figure.

   Figure 15: Port Breakout Mode



2. **BCM957608-N1400 devices only**: The **Enable MH Breakout** option (when enabled) allows a physical port to be divided into multiple logical ports based on the number of hosts. For example, if there are two hosts, there will be two logical ports.
3. Reboot the server for this change to take effect.

Linux Configuration for Port Breakout

The Linux `niccli` tools can be used to configure Port Breakout with the following commands:

NICCLI Commands for Port Breakout 1 port to 2

```
niccli -i <index> nvm --setoption port_operation_mode --value 8
```

A system reboot is required for this change to take effect.

NICCLI Commands for Port Breakout 1 port to 4

```
niccli -i <index> nvm --setoption port_operation_mode --value 9
```

A system reboot is required for this change to take effect.

LED Port Breakout Functionality

In port breakout mode, the port LEDs are modified to behave as follows:

- Traffic/Activity LED for the physical port blinks for traffic on any broken out port.
- Link/Speed LED for the physical port operates according to the broken out ports:
  - Off if no broken out port is linked up.
  - Green if all broken out ports are linked up and running at their maximum speed.
  - Amber for the following cases:
    - At least one broken out port is linked up, but not all broken out ports are linked up.
    - All broken out ports are linked up, but not all ports are operating at their maximum speed.

Configuring 200G Link Speeds on 2 x 100G BCM957508 Ethernet Network Adapters

Provides information for manually configuring a dual-port 200G (2 x 100G) BCM957508 Ethernet adapter as a single port 200G adapter (1 x 200).

The BCM957508 network adapter supports 200 gigabit per second link speed using QSFP56 optical transceivers or direct attach copper (DAC) cables. By default, 200G speed is disabled, allowing the more common 2 x 100G configuration to operate as expected. To use 200G speeds, the BCM957508 network adapter must be updated and special care must be taken with the configuration of the network switch to enable the required PAM-4 modulation parameters and Forward Error Correction (FEC) mode.

By default, the BCM957508 network adapter is configured as a 2-port device, supporting up to 100G on each port. To use 200G speed, the device must be reconfigured as a 1-port device and the 200G link speed must be enabled.

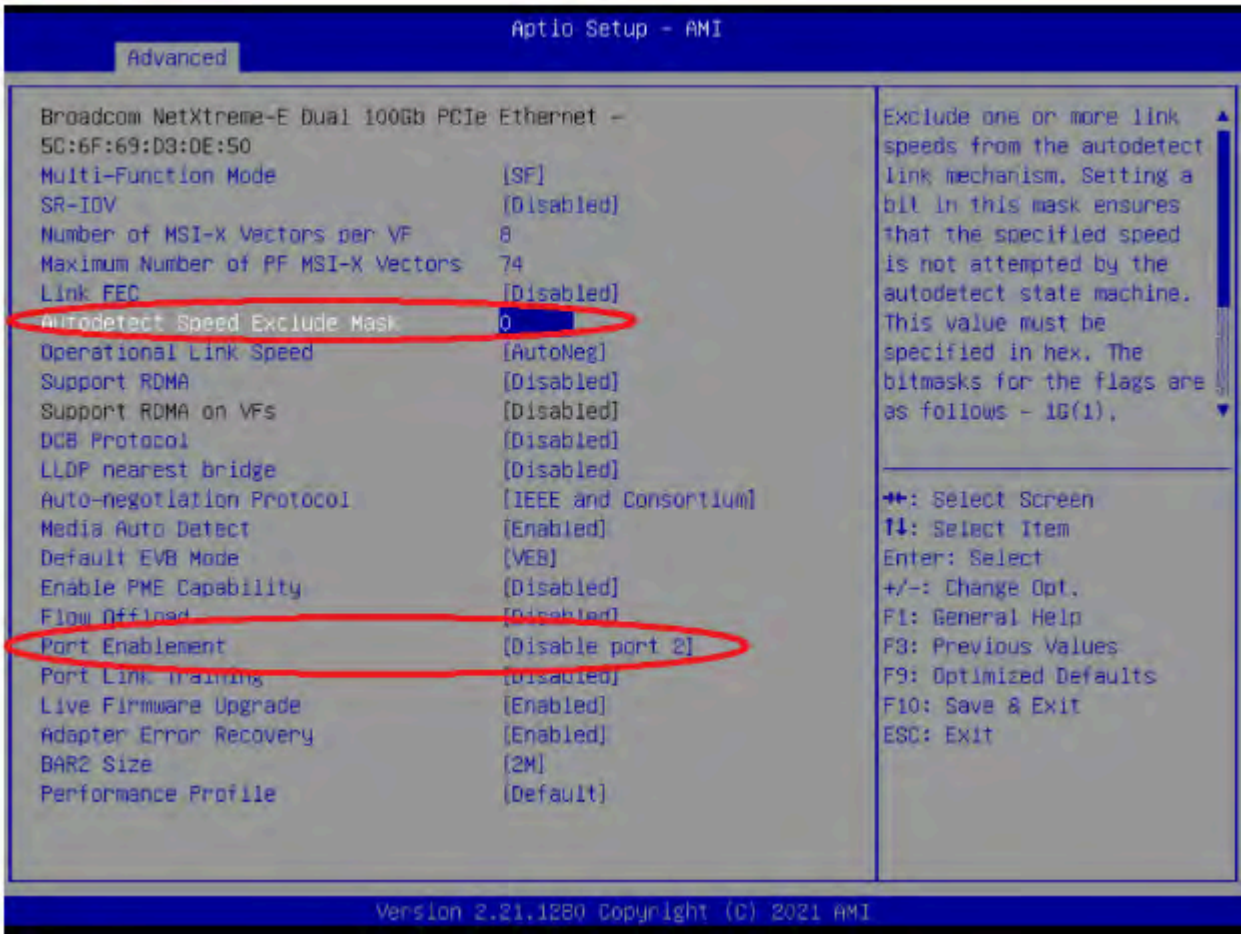
Auto-Negotiation Configuration for 200G

Provides information for auto-negotiating 200G link speeds on Ethernet network adapters.

UEFI Configuration

1. Set the Autodetect Speed Exclude Mask to 0.

2. Set Port Enablement to Disable port 2 as shown in the following figure.



3. Refer to the following table for Link FEC settings:

Table 31: Link FEC Settings

Speed	Setting
NRZ	CL91 is required
PAM-4 50G/100G	RS544 1xN
PAM-4 200G/400G	RS544 2xN

4. Refer to the following table for Link Training settings:

Table 32: Link Training Configuration per Cable Type

Cable Type	Link Training Enable/Disable
Direct Attach Copper (DAC)	Link Training – Enabled
Active Optical Cable (AOC)	Link Training – Disabled
Active Electrical Cable (AEC)	Link Training – Disabled
Active Copper Cable (ACC)	Link Training – Enabled

Cable Type	Link Training Enable/Disable
Linear Drive Pluggable Optics (LPO)	Link Training – Disabled

Linux Configuration

The Linux `niccli` tools can be used to hide the 2nd network port and include 200G link speed for autodetect to work with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption port_hide --value 1
niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope 0 --value 0x0
```

A system reboot is required for this change to take effect.

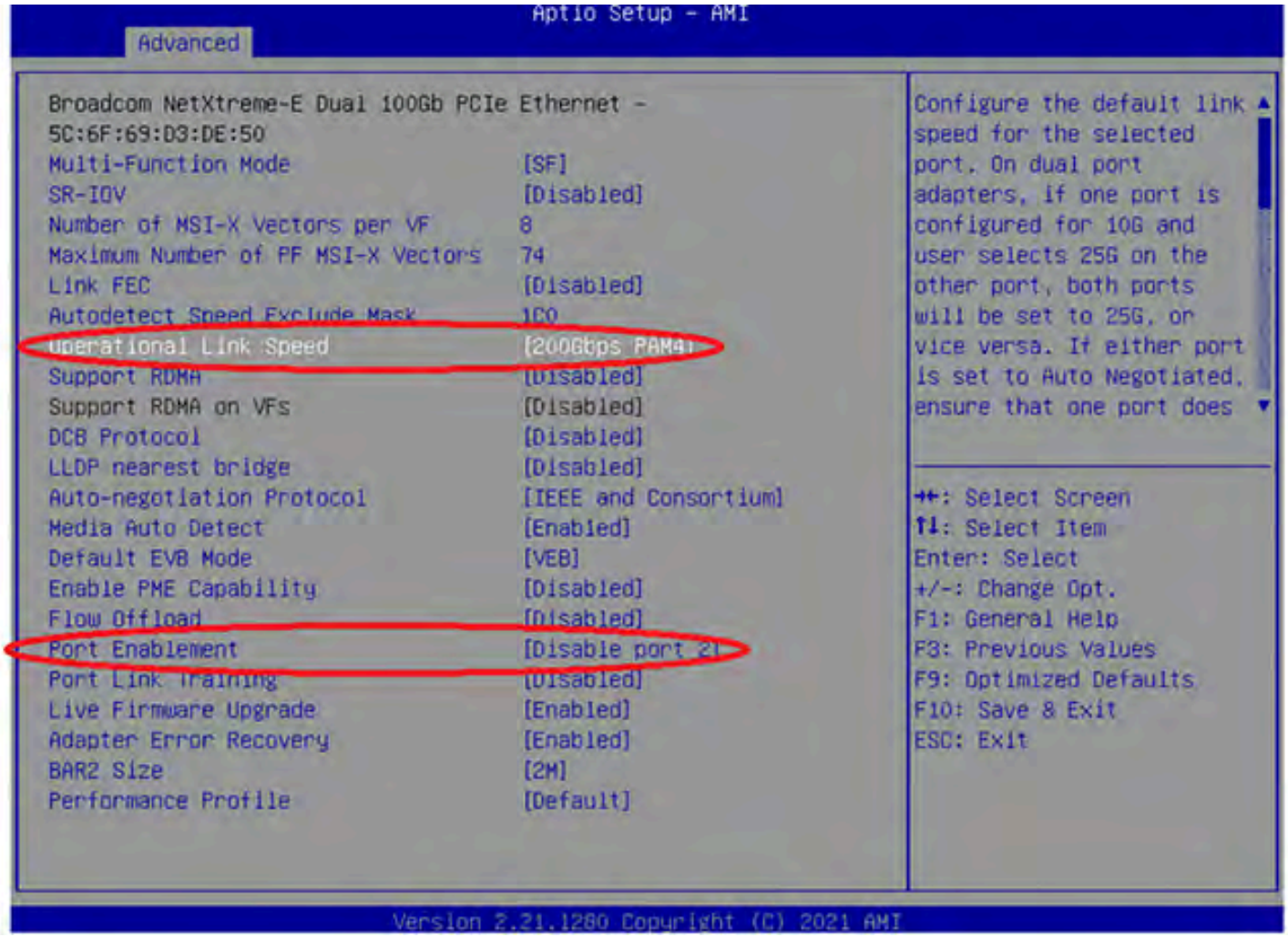
Forced 200G Configuration

Provides information for manually configuring 200G link speeds on Ethernet network adapters.

UEFI Configuration

To reconfigure the BCM957508 network adapter as a 1-port, 200G device using the UEFI:

1. Set the port speed to 200Gbps PAM-4.
2. Set Port Enablement to Disable port 2 as shown in the following figure.



Linux Configuration

The Linux `niccli` tools can be used to set the default link speed to 200G for both the driver (when the OS is running) and the firmware (PXE boot), and hide the 2nd network port with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption firmware_link_speed_d0 --scope 0 --value 0x07
niccli -i <index> nvm --setoption port_hide --value 1
```

Configuring Arista Switches

When using an Arista switch for 200G operation with DAC cables, the default interface configuration must be changed to enable Link Training and Forward Error Correction. When using 200G optical transceivers, only the speed must be set.

```
enable
configure
interface Ethernet <Port/Sub>
speed forced 200g
```

Configuring Dell Switches

When using a Dell OS10 switch for 200G operation with DAC cables, the default interface configuration must be changed to set the port group profile to 200x2, enable Link Training, and Forward Error Correction. When using 200G optical transceivers only the speed must be set.

OS10

Note: OS10 version 10.5.3.4 or later must be used.

```
enable
configure
port-group <Group ID>
port <Port ID> mode Eth 200g-2x
exit

interface range ethernet <Port ID>:1-ethernet<Port ID>:5
fec CL119-RS
negotiation off
```

SoNic

SONiC Version 4.2 or Higher

```
sonic-cli
configure terminal
interface breakout port <slot/port> mode 1x200G
interface ethernet <slot/port[/subport]>
Standalone-link-training
```

Configuring 200G Link Speeds on 2 x 100G BCM957608 Ethernet Network Adapters

Provides information for manually configuring a dual-port 200G (2 x 100G) BCM957608 Ethernet adapter as a single port 200G adapter (1 x 200).

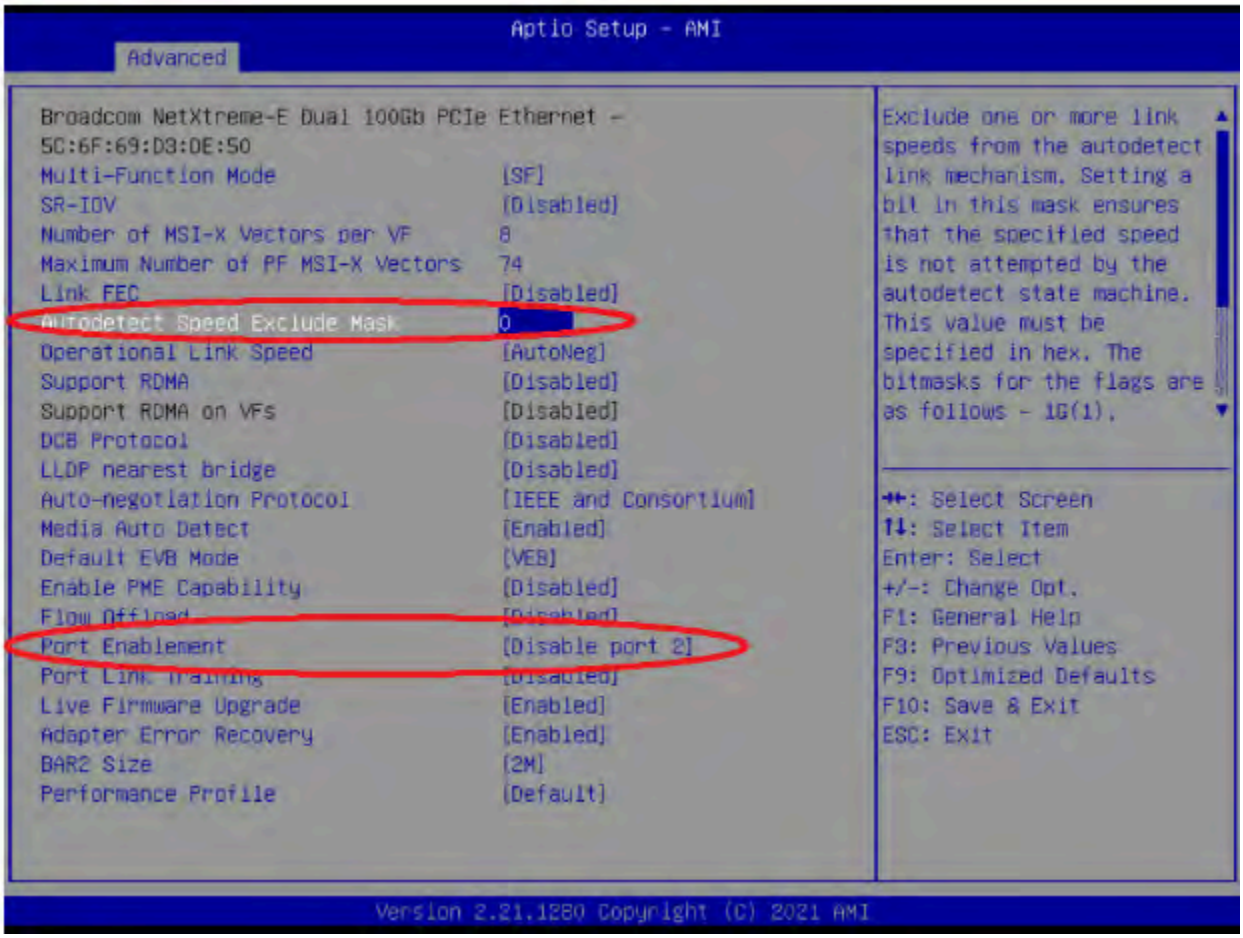
The BCM57608 network adapter supports 200 gigabit per second link speeds using QSFP112 optical transceivers or direct attach copper (DAC) cables. By default, 200G speed is disabled, allowing the more common 2 x 100G configuration to operate as expected. To use 200G speeds, the BCM957608 network adapter must be updated and special care must be taken with the configuration of the network switch to enable the required PAM-4 modulation parameters and Forward Error Correction (FEC) mode. By default, the BCM957608 network adapter is configured as a 2-port device, supporting up to 100G on each port. To use 200G speed, the device must be reconfigured as a 1-port device and the 200G link speed must be enabled.

Auto-Negotiation Configuration for 200G

Provides information for auto-negotiating 200G link speeds on Ethernet network adapters.

UEFI Configuration

1. Set the Autodetect Speed Exclude Mask to 0.
2. Set Port Enablement to Disable port 2 as shown in the following figure.



Linux Configuration

The Linux `niccli` tools can be used to hide the 2nd network port and include 200G link speed for autodetect to work with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption port_operation_mode --value 4
niccli -i <index> nvm --setoption autodetect_speed_exclude_mask --scope 0 --value 0x0
```

A system reboot is required for this change to take effect.

Forced 200G Configuration

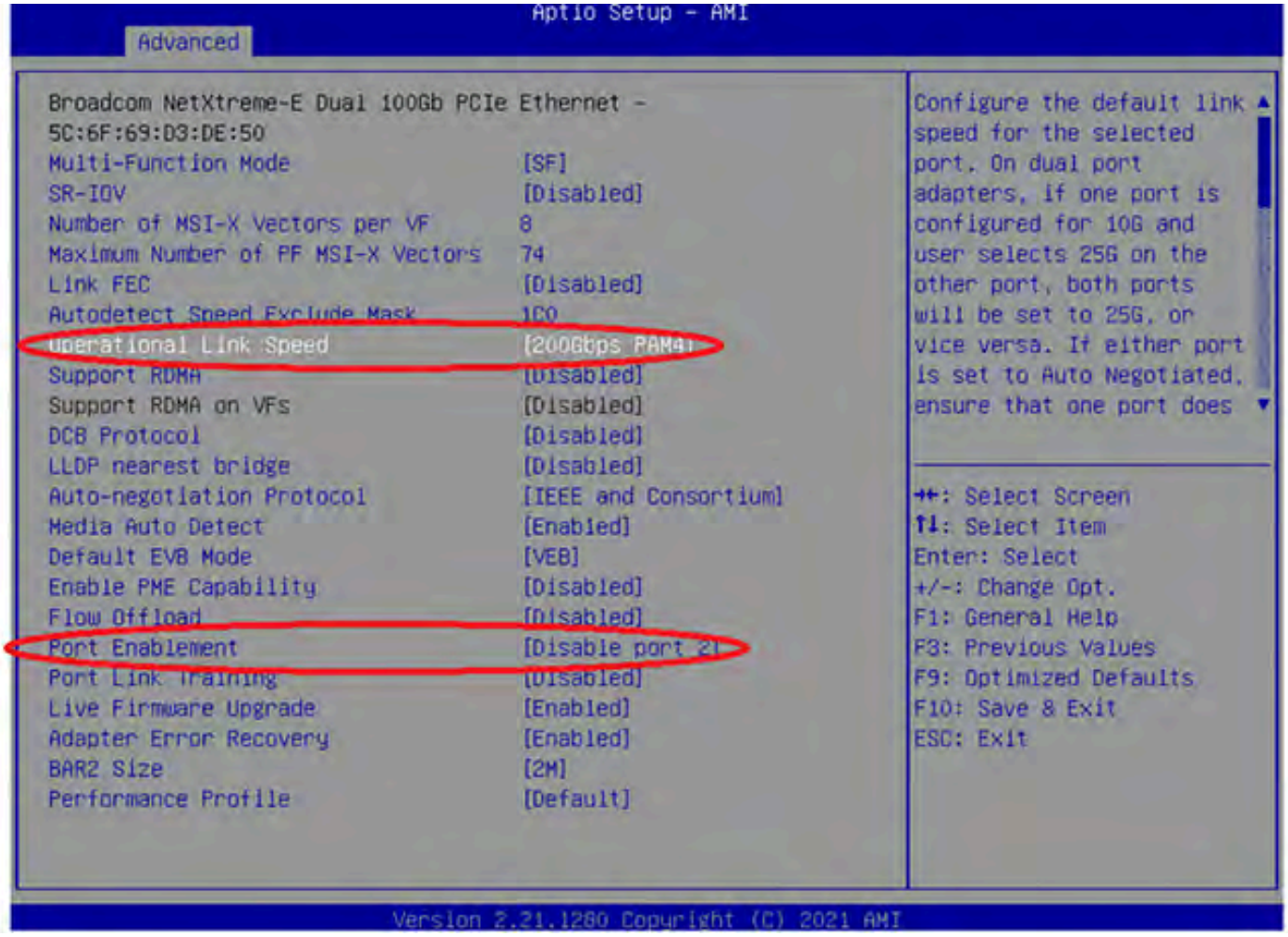
Provides information for manually configuring 200G link speeds on Ethernet network adapters.

UEFI Configuration

To reconfigure the BCM957608 network adapter as a 1-port, 200G device using the UEFI:

1. Set the port speed to 200Gbps PAM-4.

2. Set Port Enablement to Disable port 2 as shown in the following figure.



3. Refer to the following table for Link FEC settings:

Table 33: Link FEC Modes and Devices

FEC Modes	Devices
CL74 – Fire Code	BCM95741X, BCM95750X (For BCM95750X devices, FEC CL74 mode is supported for 25G link speeds only.)
CL91 – Reed Solomon	BCM957416, BCM95750X, BCM957608 (FEC CL91 mode is supported for 25G NRZ, 50G NRZ, and 100G NRZ speeds only.)
RS544 – RS544, using 1 x N RS	BCM95750X, BCM957608 (FEC RS544_1xN mode is supported for 50G PAM-4 and 100G PAM-4 speeds only.)
RS272 – RS272, using 1 x N RS	BCM95750X
RS544 – RS544, using 2 x N RS	BCM95750X, BCM95760X (FEC RS544_2xN mode is supported for 200G PAM4 speed and up.)
RS272 – RS272, using 2 x N RS	BCM95750X

4. Refer to the following table for Link Training settings:

Table 34: Link Training Configuration per Cable Type

Cable Type	Link Training Enable/Disable
Direct Attach Copper (DAC)	Link Training – Enabled
Active Optical Cable (AOC)	Link Training – Disabled
Active Electrical Cable (AEC)	Link Training – Disabled
Active Copper Cable (ACC)	Link Training – Enabled
Linear Drive Pluggable Optics (LPO)	Link Training – Disabled

Linux Configuration

The Linux `niccli` tools can be used to set the default link speed to 200G for both the driver (when the OS is running) and the firmware (PXE boot), and hide the 2nd network port with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption firmware_link_speed_d0 --scope 0 --value 0x07
niccli -i <index> nvm --setoption port_operation_mode --value 4
```

Configuring Arista Switches

When using an Arista switch for 200G operation with DAC cables, the default interface configuration must be changed to enable Link Training and Forward Error Correction. When using 200G optical transceivers, only the speed must be set.

```
enable
configure
interface Ethernet <Port/Sub>
speed forced 200g
```

Configuring Dell Switches

When using a Dell OS10 switch for 200G operation with DAC cables, the default interface configuration must be changed to set the port group profile to 200x2, enable Link Training, and Forward Error Correction. When using 200G optical transceivers only the speed must be set.

OS10

Note: OS10 version 10.5.3.4 or later must be used.

```
enable
configure
port-group <Group ID>
port <Port ID> mode Eth 200g-2x
exit

interface range ethernet <Port ID>:1-ethernet<Port ID>:5
fec CL119-RS
negotiation off
```

SoNic

SONiC Version 4.2 or Higher

```
sonic-cli
configure terminal
interface breakout port <slot/port> mode 1x200G
interface ethernet <slot/port[/subport]>
Standalone-link-training
```

Configuring 200G and 400G Link Speeds on 2 x 200G BCM957608 Dual-Port Ethernet Network Adapters

Provides information for manually configuring a dual-port 400G (2 x 200G) BCM957608 Ethernet adapter and switches.

This section provides information for configuring your Ethernet network adapter for 200G and 400G link speeds.

- Default Configurations
- Configuring 200G Link Speeds
- Configuring 400G Link Speeds

Default Configurations

The BCM957608 network adapter is configured by default to auto-negotiate and auto detect the appropriate speed of the switch port and ensure a proper link:

- Speeds up to 200G – No additional configuration is required. The BCM957608 (single or dual-port) is configured by default to link up at the maximum speed supported by the switch, up to 200G.
- 400G speeds – The BCM957608 dual-port Ethernet adapter can be converted to single-port adapter and automatically detect the speed up to 400G. The **Port Operation Mode** setting must be manually configured.

Other configurations can be manually set using the information in the following sections.

Configuring 400G Link Speeds

This section provides information for manually configuring 400G link speeds on Ethernet network adapters and switches. By default, the BCM957608 dual-port Ethernet network adapter is a 2-port device, supporting up to 200G on each port simultaneously.

Port Operational Mode settings alternatively achieve 400G single port operation utilizing 400Gbps modules. Methods to configure these modes are described in the following sections:

- Configuring 400G (Disable Port 2)
- Configuring 400G (Splitter Cable)

Methods for configuring a switch for 400G operation are described in the following section:

- Configuring Dell Switches (400G)

Configuring 400G (Disable Port 2)

This section provides information for configuring 400G link speeds using **Port Operation Mode** as **Disable Port 2**.

UEFI Configuration

To disable Port 2 using the UEFI:

1. See Configuring Dell Switches (400G) for switch settings.

2. Set **Port Operation Mode** to **Disable port 2** as shown in the following figure.

Figure 16: Device Configuration Menu

3. Reboot the server for this change to take effect.

Linux Configuration

The Linux `niccli` tools can be used to disable Port 2 with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption port_operation_mode --value 4
```

A system reboot is required for this change to take effect.

Single Port Switch Configuration (4 Lanes, PAM-4-112)

SONiC Version 4.2 or Higher

```
sonic-cli
configure terminal
interface breakout port <slot/port> mode 1x400G
interface ethernet <slot/port[/subport]>
Standalone-link-training
```

Configuring 400G (Splitter Cable)

This section provides information for manually configuring 400G link speeds on Ethernet network adapters using a splitter cable.

UEFI Configuration

To manually configure 400G link speeds with a splitter cable using the UEFI:

1. See Configuring Dell Switches (400G) for switch settings.
2. Connect splitter cable to both ports on the Ethernet network adapter.
3. Set **Port Operation Mode** to **Aggregate 2 ports to 1** as shown in the following figure.

Figure 17: Port Operation Mode

4. Reboot the server for this change to take effect.

Linux Configuration

The Linux `niccli` tool can be used to configure 400G link speeds using a splitter cable with the following commands:

NICCLI Commands

```
niccli -i <index> nvm --setoption port_operation_mode --value 5
```

A system reboot is required for this change to take effect.

Aggregate Ports with Splitter Cable (8 Lanes, PAM-4-56)

SONiC Version 4.2 or Higher

```
sonic-cli
configure terminal
interface breakout port <slot/port> mode 1x400G
interface ethernet <slot/port[/subport]>
Standalone-link-training
```

Configuring Forced Link Speeds

By default, the BCM957608 dual-port is automatically configured for 200G speeds. This section provides information on configuring the adapter to force 200G PAM-4 link speeds as an example. It is also possible to configure 200G PAM-4-112, 400G PAM4, 400G PAM-4-112, and other supported speeds.

Configuring Forced 200G PAM-4 Speed Example

Provides an example for configuring forced 200G PAM-4 link speeds on Ethernet network adapters.

UEFI Configuration

To reconfigure the BCM957608 network adapter using the UEFI:

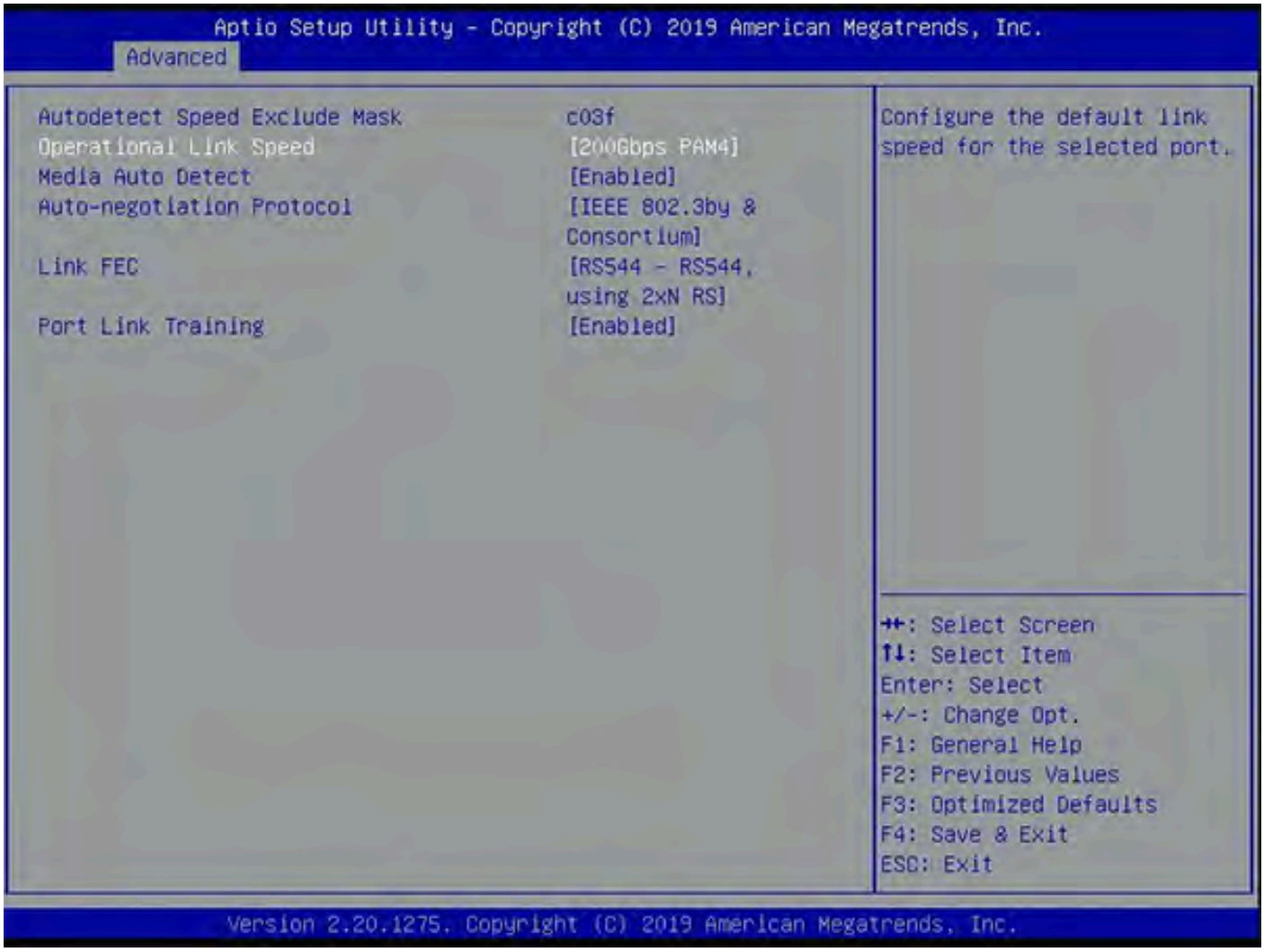
1. Set **Port Operation Mode** to **Physical Port Configuration** as shown in the following figure (default).

Figure 18: Device Configuration Menu

Note: By default, the **Media Auto Detect** feature is enabled. The following steps describe how to override this feature when required.

2. Set the **Operational Link Speed** to **200G PAM4** in the **Link Configuration** menu as shown in the following figure.

Figure 19: Link Configuration



3. Refer to the following table for Link FEC settings:

Table 35: Link FEC Settings

Speed	Setting
NRZ	CL91 is required
PAM-4 50G/100G	RS544 1xN
PAM-4 200G/400G	RS544 2xN

4. Refer to the following table for Link Training settings:

Table 36: Link Training Configuration per Cable Type

Cable Type	Link Training Enable/Disable
Direct Attach Copper (DAC)	Link Training – Enabled
Active Optical Cable (AOC)	Link Training – Disabled
Active Electrical Cable (AEC)	Link Training – Disabled
Active Copper Cable (ACC)	Link Training – Enabled
Linear Drive Pluggable Optics (LPO)	Link Training – Disabled

5. Reboot the server for this change to take effect.

Linux Configuration

The Linux `niccli` tools can be used to set the default link speed to 400G for both the driver (when the OS is running) and the firmware (PXE boot), and hide the 2nd network port with the following commands:

NICCLI Commands for Set Port Operation Mode (Example 200G PAM-4)

```
niccli -i <index> nvm --setoption port_operation_mode --value 0
```

A system reboot is required for this change to take effect.

NICCLI Commands for Set Operational Link Speed (Example 200G PAM4)

```
niccli -i <index> nvm --setoption firmware_link_speed_d0 --scope 0 --value 7
```

A system reboot is required for this change to take effect.

Note: For BCM957608, PAM-4 and NRZ speeds cannot be used on different ports of the same device simultaneously.

PXE Boot on Ethernet Network Adapters

Provides information on configuring a PXE server.

To serve PXE requests from PXE clients, a PXE server must be configured. The PXE server can be configured to run regular PXE or iPXE. Regular PXE is a network boot program that downloads config files over TFTP from the PXE server. iPXE is an enhanced implementation of the PXE client firmware and a network boot program, which uses iPXE scripts rather than config files and can download scripts and images with HTTP.

Note: UEFI mode PXE boot is supported on physical functions as well as NIC partitions, whereas legacy mode PXE boot is supported only on physical functions.

The procedure for configuring the PXE server is discussed in PXE Server Configuration.

The following sections provide information on PXE boot:

- UEFI Mode
- Legacy BIOS Mode
- PXE Boot with VLAN
- PXE Server Configuration

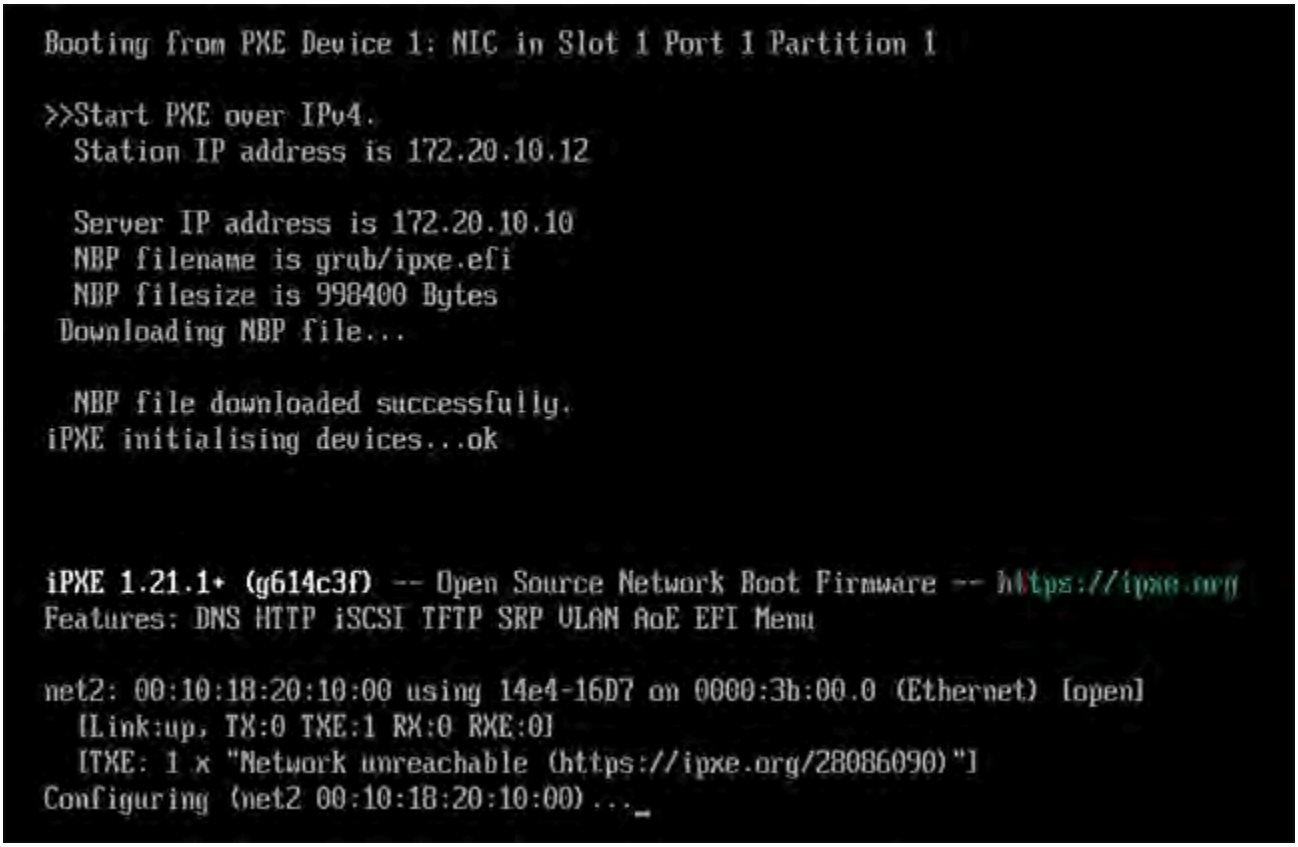
UEFI Mode

Enable PXE for the Broadcom adapter interface in **Network Configuration** under **System Setup**. Refer to the documentation from the server manufacturer on how to enable PXE on the particular server model.

The following sections provide information on UEFI Mode:

iPXE

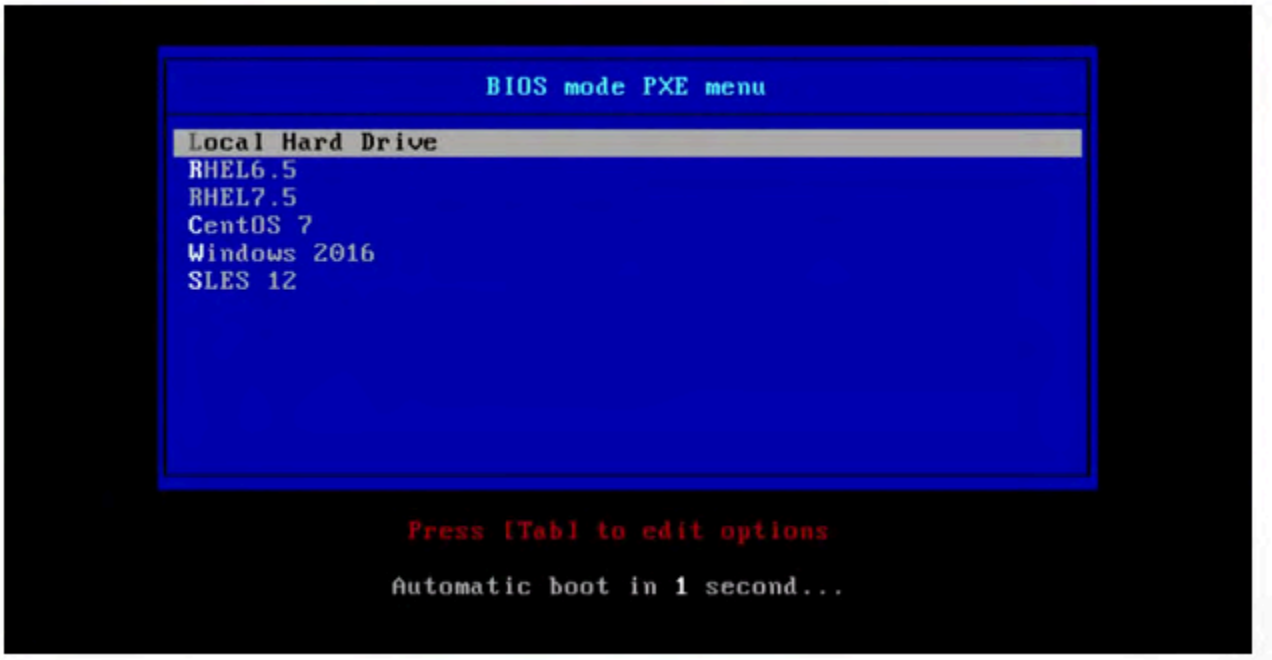
During PXE boot, the host downloads the boot load file `ipxe.pxe`, by using TFTP from the PXE server.



PXE

During PXE boot, the UEFI BIOS downloads the PXE image using TFTP from the PXE server. The PXE image brings up the menu.

The menu items from the `pxelinux.cfg/default` file are listed.



Additional files are downloaded from the PXE server over TFTP based on the menu selection and the PXE boot continues.

Legacy BIOS Mode

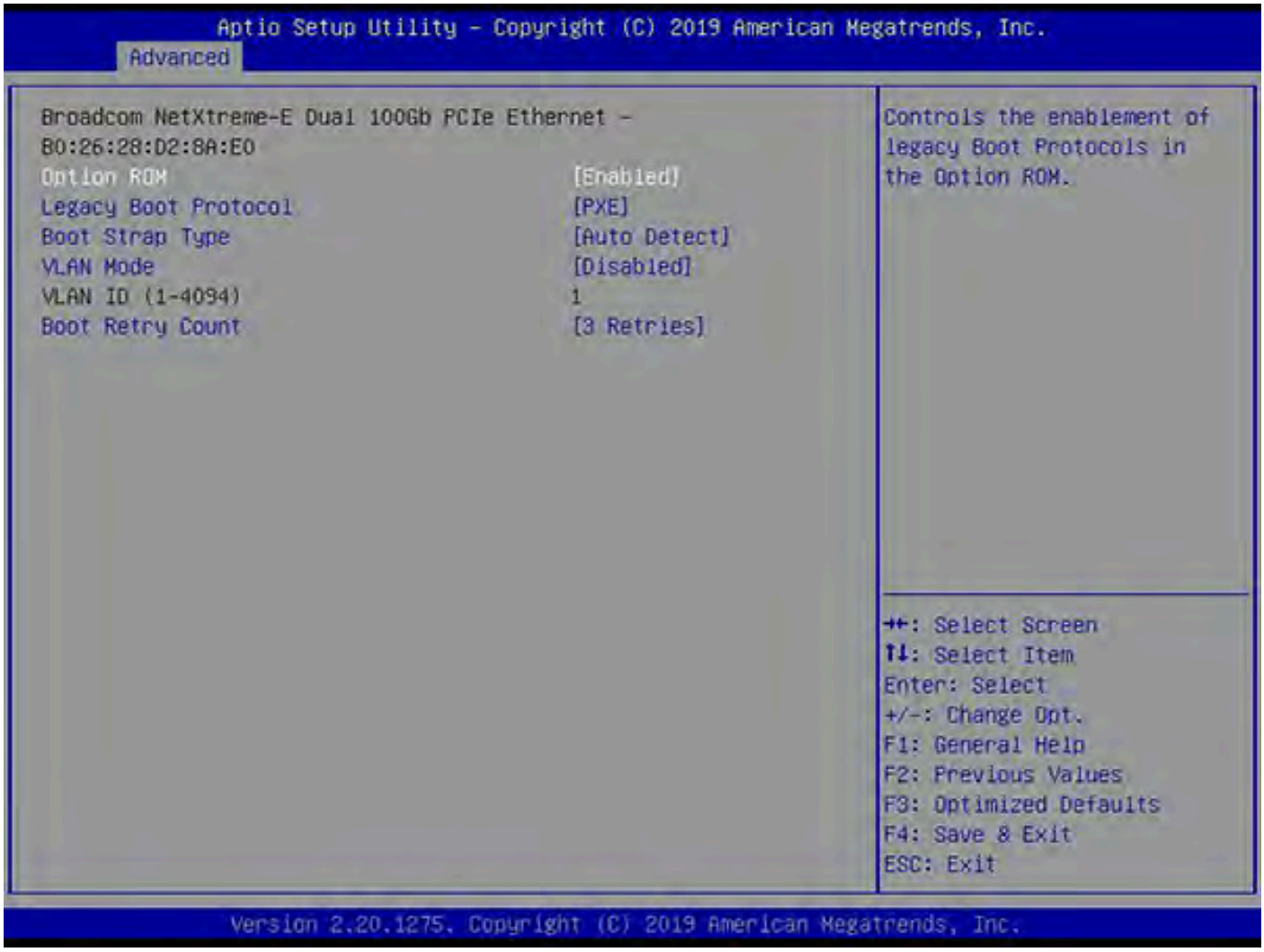
Use the following steps to configure PXE for legacy BIOS mode:

Note: For Legacy BIOS mode, only TFTP-based PXE over IPv4 is supported. Configuration for the Legacy PXE boot options is only available via the UEFI HII menus.

1. Access the UEFI HII **MBA Configuration** menu.

2. Set the **Legacy Boot Protocol** to **PXE** in the **MBA Configuration** menu.

Figure 20: MBA Configuration Menu



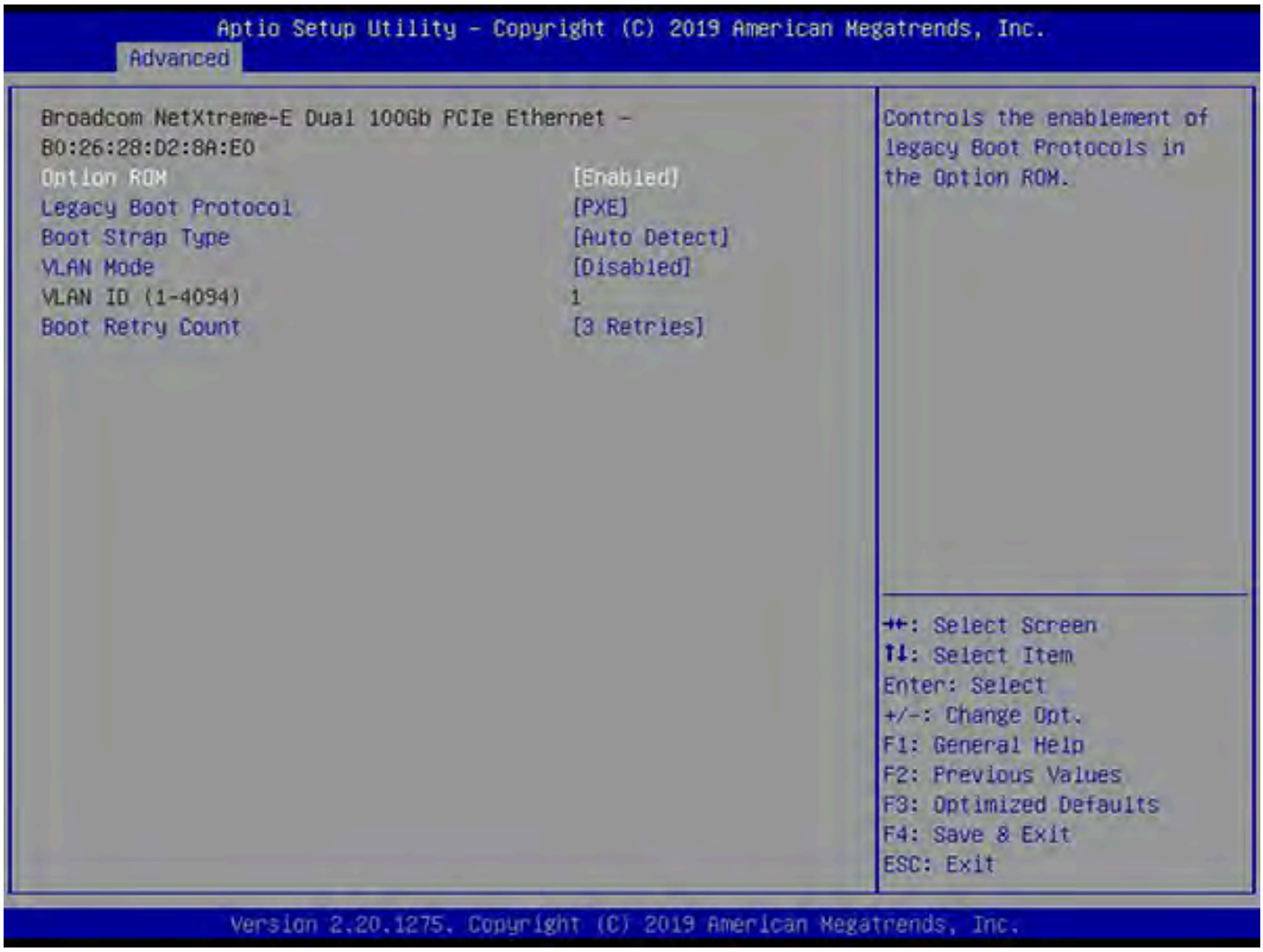
3. Save and reboot the server.

PXE Boot with VLAN

Use the following steps to configure PXE boot with VLAN:

1. Set **VLAN Mode** to **Enabled** and assign a valid Virtual LAN ID in the **MBA Configuration** menu.

Figure 21: MBA Configuration Menu



2. Configure the PXE server interface with a matching VLAN ID.
   Note: The **Virtual LAN Mode** and **Virtual LAN ID** parameters on the **NIC Configuration** menu apply to PXE boot in UEFI mode as well. If **Virtual LAN** configuration is done from both the **System Setup** menus as well as the adapter **NIC Configuration** menu, the **Virtual LAN ID** configured through **System Setup** takes precedence.

3. Trigger PXE on the adapter interface.
   After the bootloader is downloaded, press **Ctrl+B** to enter the **iPXE** menu.

4. After entering the iPXE shell, run the following commands to add a VLAN tag on the adapter interface.

- `ifstat`[to identify the adapter interface by its MAC address]
- `vcreate --tag <VLANID> <interface>`
- `autoboot <interface>`

This command sequence starts the PXE boot on the test interface with VLAN.

To destroy the VLAN tag on the adapter interface, run the `vdestroy <VLAN_interface>` command in the iPXE shell.

Note: For iPXE to support the `vcreate` command, iPXE must be built with `vlan_cmd` enabled in buildcfg. For more details, refer to https://ipxe.org/buildcfg/vlan_cmd.

PXE Server Configuration

The PXE server is operating system-independent. A PXE server is any host that can lease out DHCP IPs to any requesting PXE client, point to a boot loader file, and transfer further configuration and boot files to the PXE client over any application-level protocol.

This section explains how to set up a PXE server on a RHEL/CentOS-based operating system. For Windows deployment services, refer to the documentation provided on the Microsoft Technet website. For setting up PXE servers on any other operating system, refer to the documentation provided by the respective operating system.

The following sections provide information on PXE server configuration:

- Configuring DHCP for PXE/iPXE
- Configuring TFTP
- Configuring HTTP

Configuring DHCP for PXE/iPXE

Provides instructions for configuring DHCP on PXE and iPXE servers.

To add the DHCP feature on the PXE server, use the following command:

```
yum install dhcp
```

Note: All IP addresses mentioned in this section are for illustration purposes only. Modify the IP address to match the subnet of the PXE server configuration.

IPv4 DHCP Configuration

To configure an interface with static network settings using `ifcfg` files, for an interface with the name p1p1, create a file with name ifcfg-p1p1 in the `/etc/sysconfig/network-scripts/` directory as follows:

```
DEVICE=p1p1
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
IPADDR=174.30.10.10
NETMASK=255.255.0.0
```

1. iPXE
   To run iPXE, in the `/etc/dhcp/dhcpd.conf` file, make the following changes:

```
option ipxe.no-pxedhcp 1;
subnet 174.30.10.0 netmask 255.255.0.0 {
option routers 174.30.10.10;
range 174.30.10.11 174.30.10.50;
next-server 174.30.10.10;
option subnet-mask 255.255.0.0;
default-lease-time 3600;
max-lease-time 4800;
class "pxeclients" {
match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
if not exists ipxe.bus-id {
next-server 174.30.10.10;
if option arch = 00:06 {
filename "ipxe/ipxe-x86.efi";
} elsif option arch = 00:07 {
filename "ipxe/ipxe.efi"; # iPXE.efi built with support for Broadcom adapters
} elsif option arch = 00:00 {
filename "ipxe/ipxe.pxe"; # iPXE.pxe built with support for Broadcom adapters
}
} else {
next-server 174.30.10.10;
filename "ipxe/menu.ipxe"; # iPXE boot menu
}
}
}
```

2. PXE
   To run PXE, in the `/etc/dhcp/dhcpd.conf` file, make the following changes:

```
ddns-update-style none;
default-lease-time 600;
```

```
option space PXE;
option PXE.mtftp-ipcode 1 = ip-address;
option PXE.mtftp-cport code 2 = unsigned integer 16;
option PXE.mtftp-sport code 3 = unsigned integer 16;
option PXE.mtftp-tmout code 4 = unsigned integer 8;
option PXE.mtftp-delay code 5 = unsigned integer 8;
option arch code 93 = unsigned integer 16;
allow booting; allow bootp;
allow unknown-clients;
subnet 174.30.0.0 netmask 255.255.0.0 {
default-lease-time 600;
max-lease-time 6000;
range 174.30.10.11 174.30.10.50; #
class "pxeclients" {
match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
next-server 174.30.10.10;
if option arch = 00:06 {
filename "bootia32.efi";
} else if option arch = 00:07 {
filename "BOOTX64.EFI";
} else {
filename "pxelinux/pxelinux.0";
}
}
}
```

To bind a certain hardware MAC address to an IP in the DHCP range, add the following to the `dhcpd.conf` file.

```
host server1-adapter1-port1 {
hardware ethernet 00:0A:F7:94:F7:A4;
fixed-address 174.30.10.15;
}
```

To restart the network service, run the following command:

```
service network restart
```

To restart the DHCP service, run the following command:

```
service dhcpd restart
```

IPv6 DHCP Configuration

To enable/disable IPv6 DHCP:

In `/etc/sysconfig/network` , make the following changes:

```
NETWORKING_IPV6=yes
IPV6FORWARDING=no
IPV6_AUTOCONF=no
IPV6_AUTOTUNNEL=no
```

In `/etc/sysconfig/network-scripts/ifcfg-<interface_name>` make the following changes:

```
IPV6_AUTOCONF=no
IPV6INIT=yes
```

```
IPV6ADDR=2015:9:19:ffff::10/64 # Replace with your static address
```

1. iPXE

   In `/etc/dhcpd/dhcpd6.conf` file, make the following changes:

```
default-lease-time 2592000;
preferred-lifetime 604800;
option dhcp-renewal-time 3600;
option dhcp-rebinding-time 7200;
allow leasequery;
option dhcp6.info-refresh-time 21600;
dhcpv6-lease-file-name "/var/lib/dhcpd/dhcpd6.leases";
option dhcp6.user-class code 15 = string;
option dhcp6.bootfile-url code 59 = string;
option dhcp6.client-arch-type code 61 = array of unsigned integer 16;
option dhcp6.name-servers 2015:9:19:ffff::10;
subnet6 2015:9:19:ffff::/64 {
range6 2015:9:19:ffff::11 2015:9:19:ffff::500;
if exists dhcp6.client-arch-type and option dhcp6.client-arch-type = 00:07 {
option dhcp6.bootfile-url "tftp://[2015:9:19:ffff::10]/ipxe/ipxe.efi";
} elsif exists dhcp6.client-arch-type and option dhcp6.client-arch-type = 00:00 {
option dhcp6.bootfile-url "tftp://[ 2015:9:19:ffff::10]/pxelinux/pxelinux.0";
} elsif exists dhcp6.user-class and substring(option dhcp6.user-class, 2, 4) = "iPXE" {
option dhcp6.bootfile-url "tftp://[2015:9:19:ffff::10]/ipxe/menu.ipxe";
}
}
```

2. PXE

   In `/etc/dhcpd/dhcpd6.conf` file, make the following changes:

```
default-lease-time 2592000;
preferred-lifetime 604800;
option dhcp-renewal-time 3600;
option dhcp-rebinding-time 7200;
allow leasequery;
option dhcp6.info-refresh-time 21600;
dhcpv6-lease-file-name "/var/lib/dhcpd/dhcpd6.leases";
option dhcp6.user-class code 15 = string;
option dhcp6.bootfile-url code 59 = string;
option dhcp6.client-arch-type code 61 = array of unsigned integer 16;
option dhcp6.name-servers 2015:9:19:ffff::10;
subnet6 2015:9:19:ffff::/64 {
range6 2015:9:19:ffff::11 2015:9:19:ffff::500;
if exists dhcp6.client-arch-type and option dhcp6.client-arch-type = 00:07 {
option dhcp6.bootfile-url "tftp://[2015:9:19:ffff::10]/BOOTX64.EFI ";
} elsif exists dhcp6.client-arch-type and option dhcp6.client-arch-type = 00:00 {
option dhcp6.bootfile-url "tftp://[ 2015:9:19:ffff::10]/pxelinux/pxelinux.0";
        }
```

To restart the network service, run the following command:

```
service network restart
```

To restart the DHCPv6 service, run the following command:

```
service dhcpd6 restart
```

DHCP with VLAN

In `/etc/sysconfig/network-scripts` , make a copy of the interface file for the interface over which VLAN must be configured.

Example: ifcfg-p1p1 and ifcfg-p1p1.5

Note: 5 indicates that the interface will use VLAN ID 5.

In the new VLAN interface file, add the parameter `VLAN=yes`

Use the `uuidgen p1p1.5` command to generate a UUID for this new interface.

In the file `/etc/dhcp/dhcpd` , add a scope for the VLAN interface so that the DHCP server leases DHCP IPs on the VLAN interface.

Configuring TFTP

Provides instructions for configuring TFTP.

To configure TFTP:

1. Add the TFTP, XINET packages.
   ```
   yum install xinetd tftp-server
   ```
2. In the `/etc/xinetd.d/tftp` file, make the following changes.

```
service tftp
    {
socket_type=dgram
protocol=udp
wait=yes
user=root
server=/usr/sbin/in.tftpd
server_args=-s /var/lib/tftpboot -6
disable=no
per_source=11
cps=100 2
flags=IPv6
    }
```

The base path for the TFTP server is `/var/lib/tftpboot/`.

iPXE

Under the TFTP root directory, create a new folder called `ipxe` and add the `ipxe.efi` and `ipxe.pxe` files. Also, add the `menu.ipxe` file, which can chain more iPXE files to list the menu based on the boot mode.

```
#!ipxe
      iseq ${platform} efi && goto uefibios || goto legacybios
      :uefibios
      echo Loading the UEFI Menu
      chain --replace --autofree ${menu-url}efimenu.ipxe
      :legacybios
      echo Loading the LEGACY Menu
      chain --replace --autofree ${menu-url}biosmenu.ipxe
```

All the chained files are to be present in the `<TFTP_root>/ipxe` directory.

PXE

Create a new directory `pxelinux` in the TFTP root. Extract the Syslinux package and move the contents to the `<TFTP_root>/pxelinux` directory.

The TFTP root must contain the boot loader file from the target operating system. The boot loader file can be copied from the `/boot/ efi/EFI/<OS flavor>` directory on the target operating system. It should be renamed to `BOOTX64.EFI`

1. Create the `grub.cfg` file in the TFTP root and add UEFI mode PXE boot menu items to the same.

```
default=0 timeout=10
      title RedHat 7u5
      root (nd)
      kernel /images/rhel75/vmlinuz inst.driver=vesa nomodeset method=http://174.30.10.10/im-
ages/RHEL75linux dd
      initrd /images/RHEL7.2/initrd.img ip=dhcp
      title SLES12SP4
      root (nd)
      kernel images/sles12sp4/vmlinuz inst.driver=vesa nomodeset inst.repo=http://174.30.10.10/im-
ages/SLES12SP4
      initrd/images/sles12sp4/initrd.img ip=dhcp
```

2. Create `pxelinux.cfg/default` file and add the BIOS mode PXE boot menu items.

```
DEFAULT menu.c32
PROMPT 0
TIMEOUT 10
MENU TITLE BIOS mode PXE menu
LABEL localdisk
MENU LABEL ^Local Hard Drive LOCALBOOT 0
LABEL RHEL75
MENU LABEL ^RedHat 7u5
KERNEL images/rhel75/vmlinuz
APPEND initrd=images/rhel75/initrd.img ramdisk_size=200000 ip=dhcp inst.xdriver=vesa nomodeset inst.repo=
 http://174.30.10.10/images/RHEL75" scope="external">http://174.30.10.10/images/RHEL75
LABEL SLES12SP4
MENU LABEL ^SLES 12 SP 4
KERNEL images/sles12sp4/vmlinuz
APPEND initrd=images/sles12sp4/initrd.img ramdisk_size=200000 ip=dhcp inst.xdriver=vesa nomodeset inst.re-
po=http://174.30.10.10/images/SLES12SP4" scope="external">http://174.30.10.10/images/SLES12SP4
```

3. Create a new images directory in the TFTP root. Subdirectories can be created to match the kernel image path specified in `grub.cfg` and `pxelinux.cfg/default` files. Copy the `vmlinuz` and `initrd.img` files from the `/boot` directory of the target operating system to these subdirectories.
4. To restart the TFTP service, run the following command:

```
service xinetd restart
```

Configuring HTTP/HTTPS

Provides instructions for configuring HTTP/HTTPS.

To configure HTTP/HTTPS:

1. Add the HTTP/HTTPS feature using the following command:

```
yum install httpd
/var/www/html/
```

`Directory /var/www/html/` is the base path for the HTTP server. The configuration file is present in `/etc/httpd/conf/httpd.conf`.

2. To restart the HTTP/HTTPS service, run the following command:

   ```
   service httpd restart
   ```

   To specifically make HTTP/HTTPS listen on certain interfaces, in the `/etc/httpd/conf/httpd.conf` file, add the LISTEN directive for all the required interfaces.

   ```
   Listen 192.0.2.1:80
   Listen 192.0.2.5:8000
   Listen 174.30.10.10:80
   ```

   If this directive is not specified, the server listens on all interfaces.
3. Create a new images directory in the HTTP root. Subdirectories can be created to match the installation repository path specified in `grub.cfg` and `pxelinux.cfg/default` files. Extract the contents of the installation media of the target operating system to these subdirectories.

HTTPS setup is similar to HTTP, however, certificates must be updated in the BIOS.

SR-IOV and Use Case Examples for Ethernet Network Adapters

Provides SR-IOV configuration and use case examples for Ethernet network adapter.

This section provides the following SR-IOV configuration and use case examples:

- Enable SR-IOV in BIOS/UEFI and Device
- Linux Use Case Example: SR-IOV Pass-Through to libvirt Virtual Machine
- VMware SR-IOV Use Case Example

Enable SR-IOV in BIOS/UEFI and Device

To enable SR-IOV in BIOS/UEFI and the Ethernet adapter:

1. Enable SR-IOV in the NIC cards:
   a. SR-IOV in the NIC card can be enabled using the HII menu. During system boot, access the system **BIOS** > **NetXtreme-E NIC** > **Device Level Configuration** menu.
   b. Set the **Virtualization** mode to **SR-IOV**.
   c. Set the number of virtual functions per physical function.
   d. Set the number of MSI-X vectors per the VF and maximum number of physical function MSI-X vectors. If the VF is running out of resources, balance the number of MSI-X vectors per VM using the UEFI. Keep the default value to achieve the best resource allocation.
2. Enable virtualization in the BIOS:
   a. During system boot, enter the system **BIOS** > **Processor settings** > **Virtualization Technologies** and set it to **Enabled**.
   b. During system boot, enter the system **BIOS** > **SR-IOV Global** and set it to **Enabled**.

Linux Use Case Example: SR-IOV Pass-Through to libvirt Virtual Machine

1. Install the desired Linux version with Virtualization enabled (libvirt and Qemu).
2. Enable the IOMMU kernel parameter.
   a. The IOMMU kernel parameter is set by appending `intel_iommu=on` to the kernel command line:
   `vi /etc/default/grub (add "intel_iommu=on (Intel only) or “iommu=on” (AMD only) to GRUB_CMDLINE_LINUX`
   `grub2-mkconfig -o /boot/grub2/grub.cfg` (or `/boot/efi/<system type>/grub.cfg`)
3. Use the inbox driver, or install the driver as shown in Installing the Linux Driver.

4. Enable virtual functions through kernel parameters:
   a. After the driver is installed, lspci displays the Broadcom Ethernet NIC controller physical interfaces present in the system.
   b. To activate virtual functions, use the following command: `echo X > /sys/class/net/<ifname>/device/sriov_numvfs`

   Note: Ensure that the physical interface (<interface>) is up. VFs are only created if PFs are up. X is the number of VFs that are exported to the operating system.

   Example: `echo 4 > /sys/class/net/eth1/device/sriov_numvfs`
5. Check that the PCIe virtual functions exist with lspci.
6. Create a new virtual machine from the installation ISO file.
7. Select Customize configuration before installation.
8. Select **Add Hardware →PCI Host Device→ Virtual Function**.
9. Finish the installation. The VM detects the Broadcom Ethernet NIC controller. Use either the inbox driver from the installed operating system or the install driver as shown in Installing the Linux Driver.

VMware SR-IOV Use Case Example

1. On ESXi, install the driver as shown in Installing the VMware Driver.
2. Enable SR-IOV VFs:

Only the physical functions (PFs) are automatically enabled. If a PF supports SR-IOV, the PF(vmknicX) is part of the output of the following command.

esxcli network sriovnic list

To enable one or more virtual functions (VFs), the driver uses the module parameter `max_vfs` to enable the desired number of VFs per PF. For example, to enable four VFs on PF1:

`esxcfg-module -s 'max_vfs=4' bnxtnet` (reboot required)

To enable VFs on a set of PFs, use the following command format. The example enables four VFs on PF 0 and 2 VFs on PF 2:

`esxcfg-module -s 'max_vfs=4,2' bnxtnet` (reboot required)

The required VFs of each supported PF are enabled in order during the PF bring up. Refer to the VMware documentation for information on how to map a VF to a VM.

Note: When using NPAR + SR-IOV, every NPAR function (PF) is assigned a maximum of eight VFs.

Note: For a VF to be in promiscuous mode, one of the following conditions must be true:

- The VF should be associated with a default VLAN.
- The VF should be a trusted VF.

If none of the above conditions are true, the VF will not be in promiscuous mode and, therefore, will not see packets received by the PF.

Table 37: Recommended Configuration (BCM5750X Only)

	PCIe Physical Functions	VFs per PF	Max MSI-X Vector per PF (absolute maximum)	MSI-X Vector per VF
Single-Port Adapter	1	128	240	8
Dual-Port Adapter	2	64	120	8
Quad-Port Adapter	4	32	60	8

	PCIe Physical Functions	VFs per PF	Max MSI-X Vector per PF (absolute maximum)	MSI-X Vector per VF
With NPAR Enabled (any port count)	8	16	94	4
With NPAREP Enabled (any port count)	16	8	47	4

RDMA SR-IOV for Ethernet Network Adapters

Provides instructions for configuring RDMA SR-IOV for Ethernet network adapters to improve performance.

Single Root Input/Output Virtualization (SR-IOV) allows virtual machines to directly access PCIe devices, improving performance. The cost of SR-IOV-enabled VMs is often some loss of security and loss of live migration. The following resources are available for RDMA SR-IOV:

Table 38: RDMA SR-IOV Resources (BCM95741X and BCM95750X)

Driver/FW Version	Max. RDMA VFs	Max. aggregate QPs	RDMA SR-IOV+NPAR
218.x	0	0	No
219.x and later	128 (singlehost) 48 (multiroot - BCM95750X only)	6,144	No

Table 39: RDMA SR-IOV Resources (BCM957608)

Driver/FW Version	Max. RDMA VFs	Max. aggregate QPs	RDMA SR-IOV+NPAR
232.1 and later	32	65,536	No

Enabling RDMA SR-IOV

Enable RDMA using the NICCLI tool with the following commands.

For example, to enable RDMA on PF0 VFs:

```
niccli -i <index> nvm --setoption disable_rdma_sriov --scope 0 --value 0
```

Using RDMA SR-IOV

Refer to the documentation for your hypervisor of choice for instructions to configure SR-IOV for VMs. Both IP and RoCE are supported on Broadcom RNIC Virtual Functions, so a virtual machine only needs a single VF configured. After SR-IOV is enabled on a hypervisor, VMs can be configured and used as described in the Installing Software for Ethernet Network Adapters and Configuration sections of this document. VMs should not attempt to upgrade firmware, or configure flow control or congestion control.

NPAR and Use Case Examples for Ethernet Network Adapters

Provides configuration information for NPAR and use case examples for Ethernet network adapters.

This section provides the following information on NPAR configuration and use case examples:

- Features and Requirements
- Limitations
- Configuration
- Reducing NIC Memory Consumption with NPAR

Features

NPAR offers the following features:

- Operating System/BIOS Agnostic – The partitions are presented to the operating system as real network interfaces, so no special BIOS or operating system support is required, like SR-IOV.
- Additional NIC functions without requiring additional switch ports, cabling, or PCIe expansion slots.
- Traffic Shaping – The allocation of bandwidth per partition can be controlled to limit or reserve as needed.
- Can be used in a switch-independent manner – The switch does not need any special configuration or knowledge of the NPAR enablement.
- Can be used with RoCE and SR-IOV.
- Supports stateless offloads such as LSO, TPA, RSS/TSS, and RoCE.
- Alternative Routing-ID support for greater than eight functions per physical device.

Note: On the **UEFI HII Menu** page, the Broadcom Ethernet network adapters support up to 16 PFs per device on an ARI-capable system. For a 2-port device, this means up to 8 PFs for each port.

Limitations

Note: An NPAR configuration where a team, virtual switch (simple or distributed), or a logical network switch (N-VDS) contains more than one partition from the same physical port is not supported.

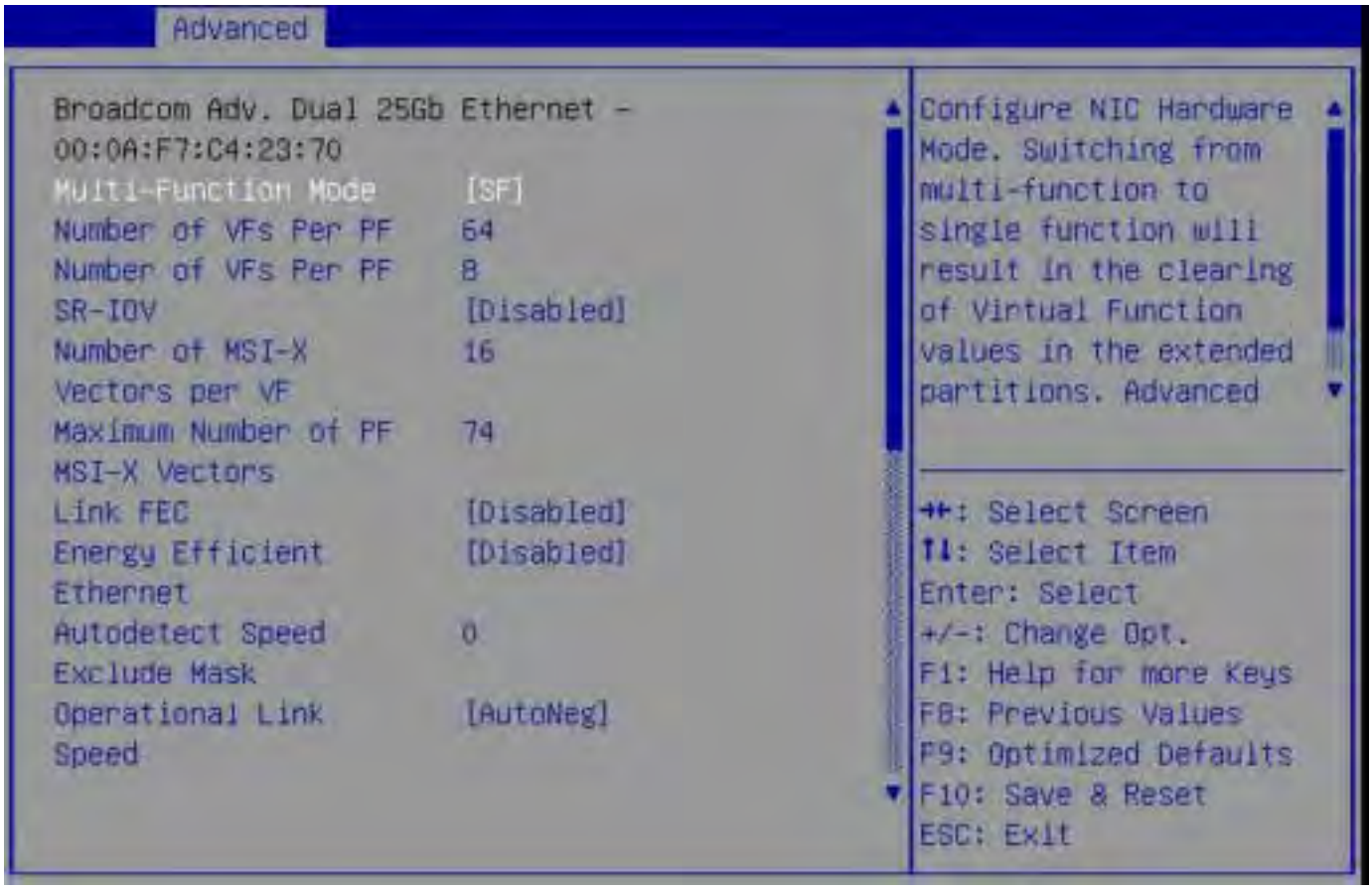
- Shared settings must be suppressed to avoid contention. For example, the device driver hides speed, duplex, flow control, and similar physical settings to avoid contention.
- Non-ARI systems enable only eight partitions per physical device.
- RoCE + SRIOV + NPAR combination is not supported for BCM5741X. It is supported for BCM95750X and BCM957608 adapters.
- RoCE for BCM5741X adapters is only supported on the first two partitions of each physical port, or a total of four partitions per physical device. BCM95750X and BCM957608 adapters can support RoCE on all partitions.
- The BCM95750X and BCM957608 support the following virtualization modes: None, NPAR, and SR-IOV.
- Teaming partitions within the same physical port are not supported.
- LACP teaming is not supported.

Configuration

NPAR can be configured using BIOS configuration HII menus or by using the Broadcom UEFI utility. Some vendors also expose the configuration via additional proprietary interfaces.

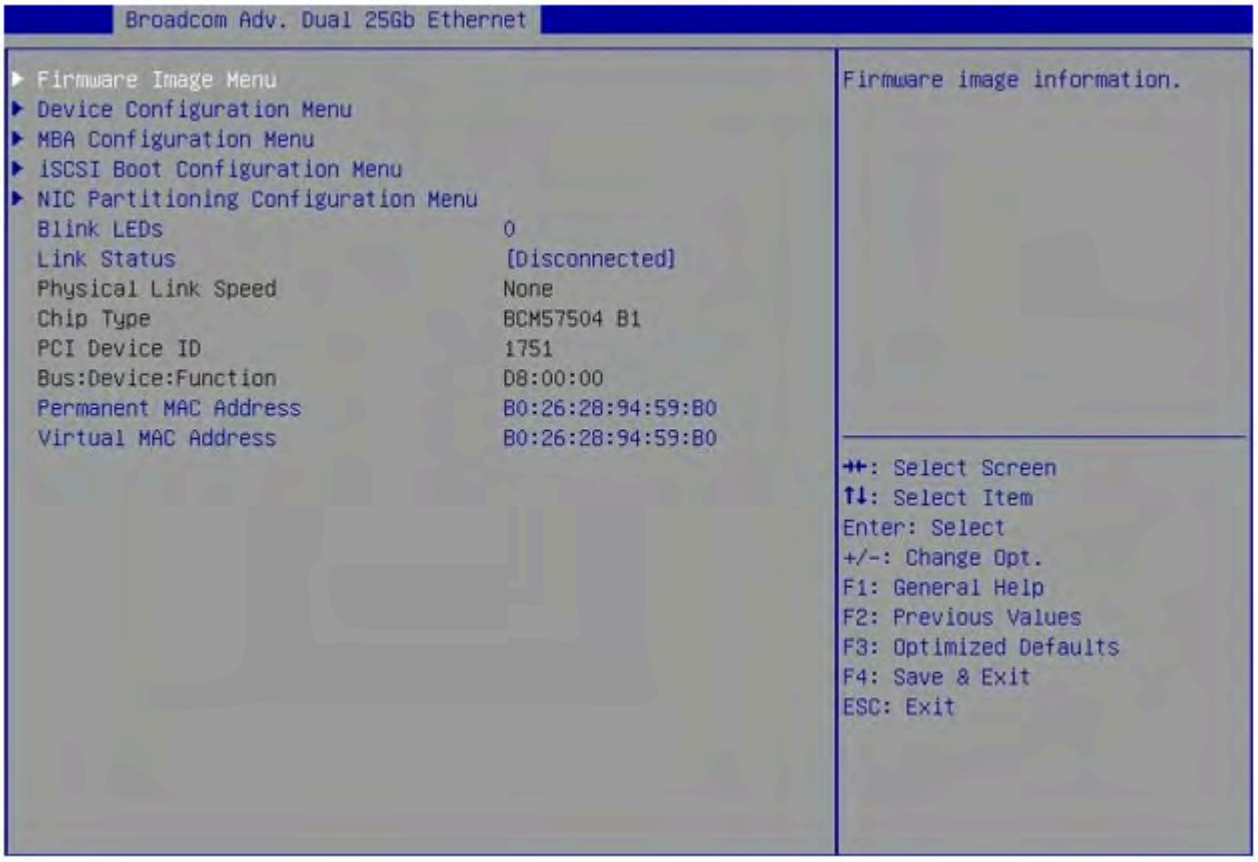
Use the following steps to enable NPAR:

1. Select the target NIC from the BIOS HII Menu and set the Multi-Function Mode or Virtualization Mode option. The choice of options affects the whole adapter instead of the individual port.



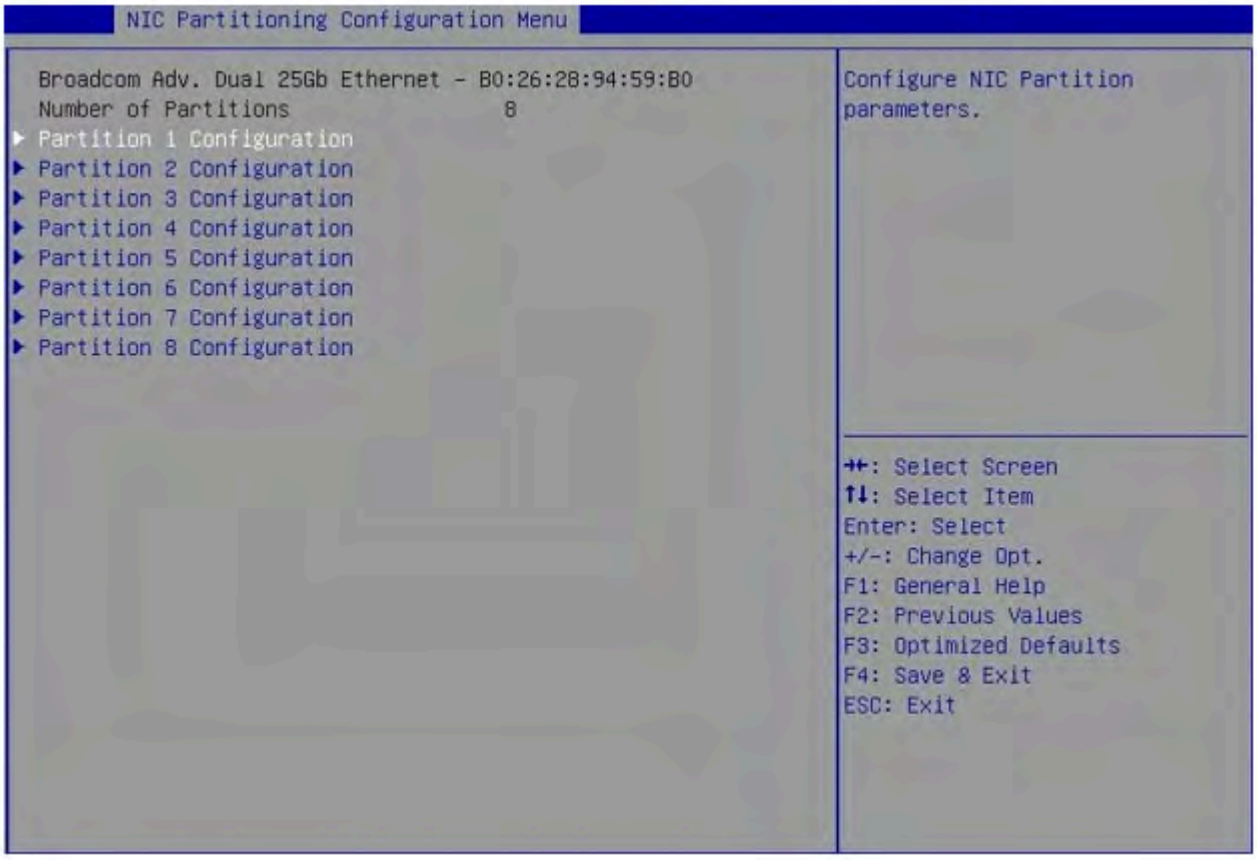
Note: For some ARI-capable OEM systems, the **NParEP** button is available to explicitly allow the adapter to support up to 16 partitions. Switching from single-function mode to multifunction mode, the device needs to be re-enumerated; therefore changes do not take effect until a system reboot occurs.

2. After NPAR is enabled, the **NIC Partitioning Main Configuration** menu option is available from the main NIC Configuration Menu associated with each physical port.



3. The NIC Partition Configuration Menu (shown in the following figure) allows the user to choose the number of partitions that should be allocated from the selected physical port. Each network adapter can support a maximum of 16 partitions on an ARI-capable server. By default, dual-port adapters are configured for eight partitions per physical port. Configuration options for each partition are also accessible from this menu. For some OEM systems, the HII menu

also includes a Global Bandwidth Allocation page where the minimum (reserved) and maximum (limit) TX bandwidth for all partitions can be configured.



4. Set the NIC Partition Configuration parameters.

Table 40: NPAR Parameters

Parameter	Description	Valid Options
BW Limit	Maximum percentage of available bandwidth this partition is allowed.	Value 0 to 100
BW Limit Valid	Functions as an on/off switch for the BW Limit setting.	True/False
RDMA Support	Functions as an on/off switch for RDMA support on this partition. **Note:** Only two partitions per physical port can support RDMA. For a dual-port device, up to 4 NPAR partitions can support RDMA.	Enabled/Disabled

Note: The NPAR minimum bandwidth parameter cannot be changed on the BCM575XX. BCM5741X devices are permitted to change the NPAR minimum value.

Reducing Network Adapter Memory Consumption with NPAR

The default value of receive buffers was selected to work well for typical configurations. If you have several adapters in a system, have enabled NPAR on multiple adapters, or have only a small amount of RAM, you might see a Code 12 yellow

error in the Device Manager for some of the adapters. Code 12 means that the driver failed to load because not enough resources exist. In this case, the resource is a specific type of kernel memory called Non-Paged Pool (NPP) memory.

If you get a Code 12, or for other reasons wish to reduce the amount of NPP memory consumed by the NIC, take the following actions:

- Reduce the number of RSS queues from the default of 8 to 4 or 2. Each RSS queue has its own set of receive buffers allocated, so reducing the number of RSS queues reduces the allocated NPP memory. There can be performance implications from reducing the number of RSS queues, as fewer cores participate in processing receive packets from that adapter. Per processor CPU utilization should be monitored to ensure that there are no hot processors after this change.
- Reduce memory allocation by reducing the number of receive buffers allocated. The default value of 0 means the driver should automatically determine the number of receive buffers. For typical configurations, a setting of 0 (=auto) maps to XXXX receive buffers per queue. You can choose a smaller value such as 1500, 1000, or 500. (The value must be a multiple of 500 and between the range of 500 and 15,000.) As previously mentioned, a smaller number of receive buffers increases the risk of packet drop and a corresponding impact on packet retransmissions and decreased throughput.

The parameters Maximum Number of RSS Queues and Receive Buffers (0=Auto) can be modified using the **Advanced** properties tab for each adapter in the **Device Manager**. If you want to modify multiple adapters at the same time, it is faster to use the `Set-NetAdapterAdvancedProperty PowerShell cmdlet`. For example, to assign two RSS queues for all adapters in a system whose adapter name starts with “Sl”, run the following command:

```
Set-NetAdapterAdvancedProperty Sl* -RegistryKeyword *NumRSSQueues -RegistryValue 2
```

Similarly, to set the number of Receive buffers to 1500, run the following command:

```
Set-NetAdapterAdvancedProperty Sl* -RegistryKeyword *ReceiveBuffers -RegistryValue 1500
```

For an overview of how to use PowerShell to modify NIC properties, refer to Microsoft.com.

DCBX – Data Center Bridging

Provides information on configuring DCBX on Ethernet network adapters.

Broadcom Ethernet network adapters support IEEE 802.1Qaz DCBX and the older CEE DCBX specification. DCB configuration is obtained by exchanging the locally configured settings with the link peer. Because the two ends of a link can be configured differently, DCBX uses a concept of *willing* to indicate which end of the link can accept parameters from the other end. This mode is indicated in the DCBX protocol by using a single bit in the ETS Configuration and PFC TLV. This bit is not used with ETS Recommendation and Application Priority TLV. By default, the controller is in willing mode while the link partner network switch is in *non-willing* mode. This default ensures that the same DCBX setting on the switch propagates to the entire network.

Users can manually set the Broadcom Ethernet network adapter to non-willing mode and perform various PFC, Strict Priority, ETS, and APP configurations from the host side. Refer to the driver `readme.txt` for additional details on available configurations. This section provides an example of how such a setting can be done in Windows with Windows PowerShell. Additional information on DCBX, QoS, and associated use cases are described in additional detail in a separate white paper, which is beyond the scope of this user guide.

The following settings in the UEFI HII menu are required to enable DCBX support:

1. Click **System** > **Setup** > **Device** > **Settings** > **NetXtreme-E** > **NIC** > **Device** > **Configuration**

This section provides the following information on DCBX:

- DCBX Mode – Enable (IEEE only)
- DCBX Willing Bit

DCBX Mode – Enable (IEEE only)

This option allows a user to enable or disable DCBX with the indicated specification. IEEE only indicates that IEEE 802.1Qaz DCBX is selected.

Windows Driver setting:

After enabling the indicated options in the UEFI HII menu to set firmware level settings, perform the following selection in the Windows driver advanced properties:

1. Open Windows Manager **Broadcom Ethernet NIC controller** > **Advanced Properties** > **Advanced tab.**
2. Quality of Service = Enabled.
3. Priority & VLAN = Priority& VLAN enabled VLAN = <ID>.
4. Set desired VLAN ID.

To exercise the DCB-related command in Windows PowerShell, install the appropriate DCB Windows feature.

1. In the **Task Bar**, right-click the Windows PowerShell icon and then click **Run as Administrator**. Windows PowerShell opens in elevated mode.
2. In the Windows PowerShell console, type:

```
Install-WindowsFeature "data-center-bridging"
```

DCBX Willing Bit

The DCBX willing bit is specified in the DCB specification. If the willing bit on a device is true, the device is willing to accept configurations from a remote device through DCBX. If the willing bit on a device is false, the device rejects any configuration from a remote device and enforces only the local configurations.

Use the following command to set the willing bit to enable (1) or disable (2). Example `set-netQoSdcbxSetting -Willing 1`

Use the following command to create a traffic class.

```
C:\> New-NetQosTrafficClass -name "SMB class" -priority 4 -bandwidthPercentage 30 -Algorithm ETS
```

Note: By default, all IEEE 802.1p values are mapped to a default traffic class, which has 100% of the bandwidth of the physical link. The previous command creates a new traffic class to which any packet tagged with eight IEEE 802.1p value 4 is mapped, and its Transmission Selection Algorithm (TSA) is ETS and has 30% of the bandwidth. It is possible to create up to seven new traffic classes. In addition to the default traffic class, there are at most eight traffic classes in the system.

Use the following command to display the created traffic class:

```
C:\> Get-NetQoSTrafficClass
NameAlgorithm Bandwidth(%)Priority
---------------------------
```

Default	ETS	70	0-3,5-7
SMB class	ETS	30	4

Use the following command to modify the Traffic Class:

```
PS C:\> Set-NetQoSTrafficClass -Name "SMB class" -BandwidthPercentage 40
PS C:\> get-NetQosTrafficClass
Name Algorithm Bandwidth(%) Priority
------------------------------------------------------------------ [Default] ETS60 0-3,5-7
SMB class ETS404
```

Use the following command to remove the traffic class:

```
PS C:\> Remove-NetQosTrafficClass -Name "SMB class"
PS C:\> Get-NetQosTrafficClass
Name Algorithm Bandwidth(%) Priority
-------------------------------------------------------------------
     [Default] ETS1000-7
```

Use the following command to create a traffic class (strict priority):

```
C:\> New-NetQosTrafficClass -name "SMB class" -priority 4 -bandwidthPercentage 30 -Algorithm Strict
```

Enabling PFC:

```
PS C:\> Enable-NetQosFlowControl -priority 4
PS C:\> Set-NetQosFlowControl -Priority 4 -Enabled $true
PS C:\> Get-NetQosFlowControl
```

Disabling PFC:

```
PS C:\> disable-NetQosflowControl -priority 4
PS C:\> Set-NetQosFlowControl -Priority 4 -Enabled $false
```

The following command creates a new policy for SMB. SMB is an inbox filter that matches TCP port 445 (reserved for SMB). If a packet is sent to TCP port 445 it is tagged by the operating system with IEEE 802.1p value of 4 before the packet is passed to a network Miniport driver. In addition to SMB, other default filters include iSCSI (matching TCP port 3260), NFS (matching TCP port 2049), LiveMigration (matching TCP port 6600), FCOE (matching EtherType 0x8906), and NetworkDirect. NetworkDirect is an abstract layer created on top of any RDMA implementation on a network adapter. NetworkDirect must be followed by a Network Direct port. In addition to the default filters, a user can classify traffic by the application's executable name (as in the first example below), or by the IP address, port, or protocol.

Use the following command to create QoS Policy:

Command:

```
PS C:\> New-NetQosPolicy -Name "SMB Policy" -SMB -PriorityValue8021Action
```

Output:

```
Name : SMB Policy
Owner : Group Policy (Machine)
NetworkProfile : All
Precedence : 127
Template : SMB
JobObject :
PriorityValue : 4
```

Use the following commands to create a QoS Policy based on the source/destination address:

Command:

```
PS C:\> New-NetQosPolicy -Name "Network Management" -IPDstPrefixMatchCondition 10.240.1.0/24 -IPProtocolMatch-
Condition Both -NetworkProfile All -PriorityValue8021Action 7
```

Output:

```
Name : Network Management
Owner : Group Policy (Machine)
Network Profile : All
Precedence : 127
IPProtocol : Both
```

```
IPDstPrefix : 10.240.1.0/24
PriorityValue : 7
```

Use the following commands to display QoS Policy:

Command:

```
PS C:\> Get-NetQosPolicy -Name "Network Management"
```

Output:

```
Owner : (382ACFAD-1E73-46BD-A0A-6-4EE0E587B95)
NetworkProfile : All Precedence : 127
IPProtocol : Both
IPDstPrefix : 10.240.1.0/24
PriorityValue : 7
Name : SMB policy
Owner : (382AFAD-1E73-46BD-A0A-6-4EE0E587B95)
NetworkProfile : All
Precedence : 127
Template : SMB
PriorityValue : 4
```

Use the following commands to modify the QoS Policy:

Command:

```
PS C:\> Set-NetQosPolicy -Name "Network Management" -IPSrcPrefixMatchCondition 10.235.2.0/24 -IPProtocolMatch-
Condition Both -PriorityValue8021Action 7
```

Output:

```
Name : Network Management
Owner : Group Policy (Machine)
NetworkProfile : All
Precedence : 127
JobObject :
IPProtocol : Both
IPSrcPrefix : 10.235.2.0/24
IPDstPrefix : 10.240.1.0/24
PriorityValue : 7
```

Configuring Peer Memory Direct with BCM95750X/BCM957608 Network Adapters

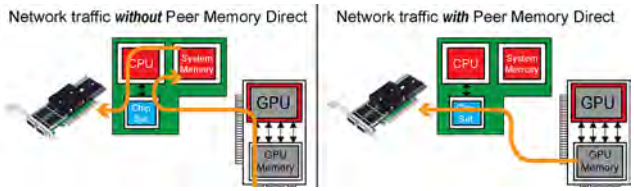
Provides information on configuring BCM95750X/BCM957608 network adapters to utilize NCCL/RCCL and Peer Memory Direct for improved performance.

Introduction

This section describes the advantages and use of Broadcom Peer Memory Direct technology. Broadcom developed Peer Memory Direct for our performance network adapters along with industry partners, standards organizations, and our customers to take maximum advantage of Graphical Processing Units (GPUs) and other accelerator devices in hyperscale data centers and High Performance Computing (HPC) installations. Artificial Intelligence/Machine Learning (AI/ML) and HPC workloads are increasingly shifting to GPUs and purpose-built accelerators to improve performance and cost, but accelerators require vast quantities of data input. In a deployment without Peer Memory Direct, this data must first be loaded by the system CPU into DRAM and then copied into the accelerator, increasing load and power draw

on the CPU, increasing latency, and causing contention in the DRAM and PCIe busses, which increases jitter (see the following figure). Peer Memory Direct allows data to be loaded into accelerators from network stores directly, without an extra copy step through DRAM with minimal latency, jitter, and power (see the following figure).

Figure 22: Peer Memory Direct



To achieve the maximum performance of Peer Memory Direct, host servers should be selected based on their PCIe topology. The GPU and the Ethernet network adapter should be able to perform Peer-to-Peer PCIe transfers without the transfers going through the PCIe root complex. Therefore, if the Ethernet network adapter and the GPU are behind the same PCIe switch, and have PCIe Access Control Services (ACS) disabled, they can engage in Peer-to-Peer transfers.

Table 41: Broadcom RNIC Part Numbers

Part Number	Form Factor	ASIC	Ports	Connector
BCM957508-P1200G	PCIe	BCM957508	1x 200G	QSFP56
BCM957508-N1200G	OCP3.0	BCM957508	1x 200G	QSFP56
BCM957508-P2100G	PCIe	BCM957508	1x 200G 2x 100G	QSFP56
BCM957508-N2100G	OCP3.0	BCM957508	1x 200G 2x 100G	QSFP56
BCM957504-N1100G	OCP3.0	BCM957504	1x 100G	QSFP56
BCM957504-N1100GD	OCP3.0	BCM957504	1x 100G	DSFP
BCM957504-P425G	PCIe	BCM957504	4x 25G	SFP28
BCM957504-N425G	OCP3.0	BCM957504	4x 25G	SFP28
BCM957608-P2200G	PCIe	BCM957608	2x 200G	QSFP112
BCM957608-N2200G	OCP3.0	BCM957608	2x 200G	QSFP112
BCM957608-P1400GD	PCIe	BCM957608	1x 400G	QSFP112
BCM957608-N1400GD	OCP3.0	BCM957608	1x 400G	QSFP112

PCIe Slot Selection for GPU and RNIC on a Host

To get the best performance for Peer Memory Direct, the PCIe slot selection for the GPU and the RNIC on a host is essential. Peer Memory Direct works via PCIe peer-to-peer transfers where data is directly transferred between the GPU and the RNIC over the PCIe bus. For PCIe peer-to-peer transfers to operate, the GPU and the RNIC should be connected to the same PCIe switch and the PCIe Access Control Services (ACS) must be disabled on the PCIe switch. If this is not accomplished, the data is transferred via the CPU root complex, and the benefit of Peer Memory Direct is reduced.

The host server PCIe topology is important since every server is not designed for Peer Memory Direct. Therefore, the host server for Peer Memory Direct must be selected carefully.

GPU Types

The following GPU types are supported by peer memory direct:

- Ethernet Networking Guide for NVIDIA H100 GPU Clusters
- Ethernet Networking Guide for AMD Instinct MI300X GPU Clusters

Windows Configuration Information

Provides information on configuring various Ethernet Adapter features in the Windows operating system.

The following list provides common Windows configuration options:

- Supported Operating Systems for Ethernet Network Adapters
- Installing the Windows Driver on Ethernet Network Adapters
- Updating the Firmware on Windows
- Installing the NICCLI Configuration Utility
- Windows RoCE Configuration
- Configuring RSS for Performance in Windows
- Link Aggregation on Ethernet Network Adapters
- Windows Statistics
- Performance Counters

ESXi Configuration Information

Provides information on configuring various Ethernet Adapter features in the ESXi operating system.

The following list provides common ESXi configuration options:

- Supported Operating Systems for Ethernet Network Adapters
- Installing the VMware Driver on Ethernet Network Adapters
- Updating the Firmware Manually on Linux/ESX
- Installing the NICCLI Configuration Utility
- Configuring RDMA over Converged Ethernet (RoCE) on VMware
- Enhanced Network Stack (ENS)
- VMware SR-IOV Use Case Example

Precision Time Protocol

Provides an introduction to precision time protocol (PTP).

Note: Precision Time Protocol is not supported on BCM9574XX devices.

Precision Time Protocol (PTP) is a protocol that is used to synchronize the clocks of devices on a network to a high level of accuracy. It is based on the Internet Protocol (IP) and is designed to synchronize the time of day across multiple devices that are connected to a network.

PTP operates by using a source clock that sends out time synchronization messages to other devices on the network, which are known as sink clocks. The sink clocks then adjust their own internal clocks based on the messages received from the source clock. PTP achieves time synchronization by timestamping the messages to determine the network delay between the sending and receiving nodes.

PTP is used in a variety of applications where accurate time synchronization is important, such as in financial systems, power grids, and telecommunications networks. It is also used in scientific and industrial applications where precise time measurement is required.

Precision time protocol is divided into the following sections:

- PTP Specification
- PTP Delay Measurements
- Installing PTP
- Configuring PTP
- ptp41 Configuration File

PTP Specification

Provides information on the PTP specification.

The main standard that specifies the Precision Time Protocol (PTP) is IEEE 1588.

The IEEE 1588 standard defines the protocol and the messages used by PTP to synchronize the clocks of devices on a network. It also specifies the hardware and software requirements for implementing PTP as well as the procedures and algorithms used to ensure accurate time synchronization.

BCM5750X network controllers are compliant with the IEEE1588v2-2019/08 standard.

PTP Messages

In the PTP, several types of messages are used to synchronize the clocks of devices on a network. These messages are sent between the source clock and the sink clocks on the network.

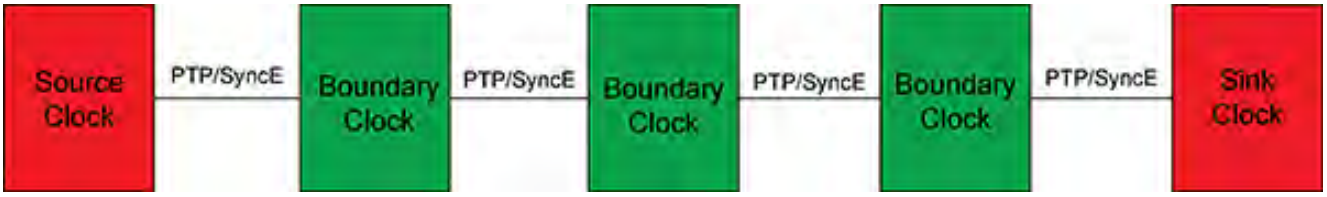
1. Event messages: Time-critical messages with an accurate timestamp that is generated at both the transmit time and receive time.
   Example event messages:
   – Sync message: This message is sent by the source clock to the sink clocks and contains the current time as measured by the source clock.
   – Follow-up message: This message is sent by the source clock in response to a sync message received from a sink clock. It contains additional information about the timing of the sync message, including the time it was sent and received.
   – Delay request message: This message is sent by a sink clock to the source clock and is used to request a delay measurement.
   – Delay response message: This message is sent by the source clock in response to a delay request message and contains the delay measurement between the source clock and the sink clock.
2. Announce message: This message is sent by the source clock and contains information about the current state of the PTP network, including the identity of the source clock and the sink clocks, as well as the current time.
3. Management message: This message is used by network management to monitor, configure, and maintain a PTP system.
4. Signaling message: This message is used to initiate or terminate PTP communication between the source clock and the sink clocks. It can also be used to request a change in the state of the PTP network, such as switching to a new source clock.

PTP System Configurations

There are three primary PTP system configurations:

- Ordinary clock – A system with one connection to the 1588v2 network. That connection/port may be a source (supplies Time of day) or sink (must be supplied time of day).
- Boundary clock: system with multiple connections - one source port and one or more sink ports.
- Transparent clock – Modifies the PTP messages as they pass through, but can correct network delays to improve the accuracy of the time distribution. This has two sub-modes: End-to-End mode and Peer-to-Peer mode.
  - End-to-End mode – In this mode, the delay is measured between the sink and the source. The source and sink send IEEE 1588 messages called DELAY REQUEST and DELAY RESPONSE between the two, allowing the delay to be measured.
  - Peer-to-Peer mode – In this mode, the delay is measured at each network element between its input port and the device attached to the other end of the wire of this input port (the peer device). As the source sends its view of time (using SYNC messages) towards sink(s), each network element along the way receives the SYNC message and adds a correction to the SYNC message. The correction includes the measured wire delay of the input port the SYNC message was received on.

Figure 23: PTP System Configurations



PTP Profiles

A PTP profile is a set of predefined configuration options for a PTP system. The PTP standard, IEEE 1588, defines several profiles that are intended for different types of applications and networks. These profiles specify different options for message timing, delay measurement, and other PTP parameters.

The main profiles defined in IEEE 1588 are:

- Default – The default profile is intended for general use and provides a balance between accuracy and network overhead.
- Power – The power profile is intended for use in power distribution systems and has a lower accuracy than the default profile.
- Telecommunication – The telecommunication profile is intended for use in telecommunication networks and has a higher accuracy than the default profile.
- High-accuracy – The high-accuracy profile is intended for use in high-accuracy applications and has the highest accuracy of all the profiles. For example, the G.8257.1 profile is considered a high-accuracy profile and is widely used in different types of networks such as mobile networks, packet-based networks, and also some specific applications such as power distribution systems and telecommunication networks.

Each profile specifies different options for message timing, delay measurement, and other PTP parameters that are tailored to the needs of a specific application or network.

The BCM5750X controller supports G.8275.1, G.8275.2, and 802.1AS profiles.

PTP Delay Measurements

Provides information on PTP delay measurements.

In PTP, a sink clock determines the time by receiving time synchronization messages from the source clock and adjusting its own internal clock based on these messages.

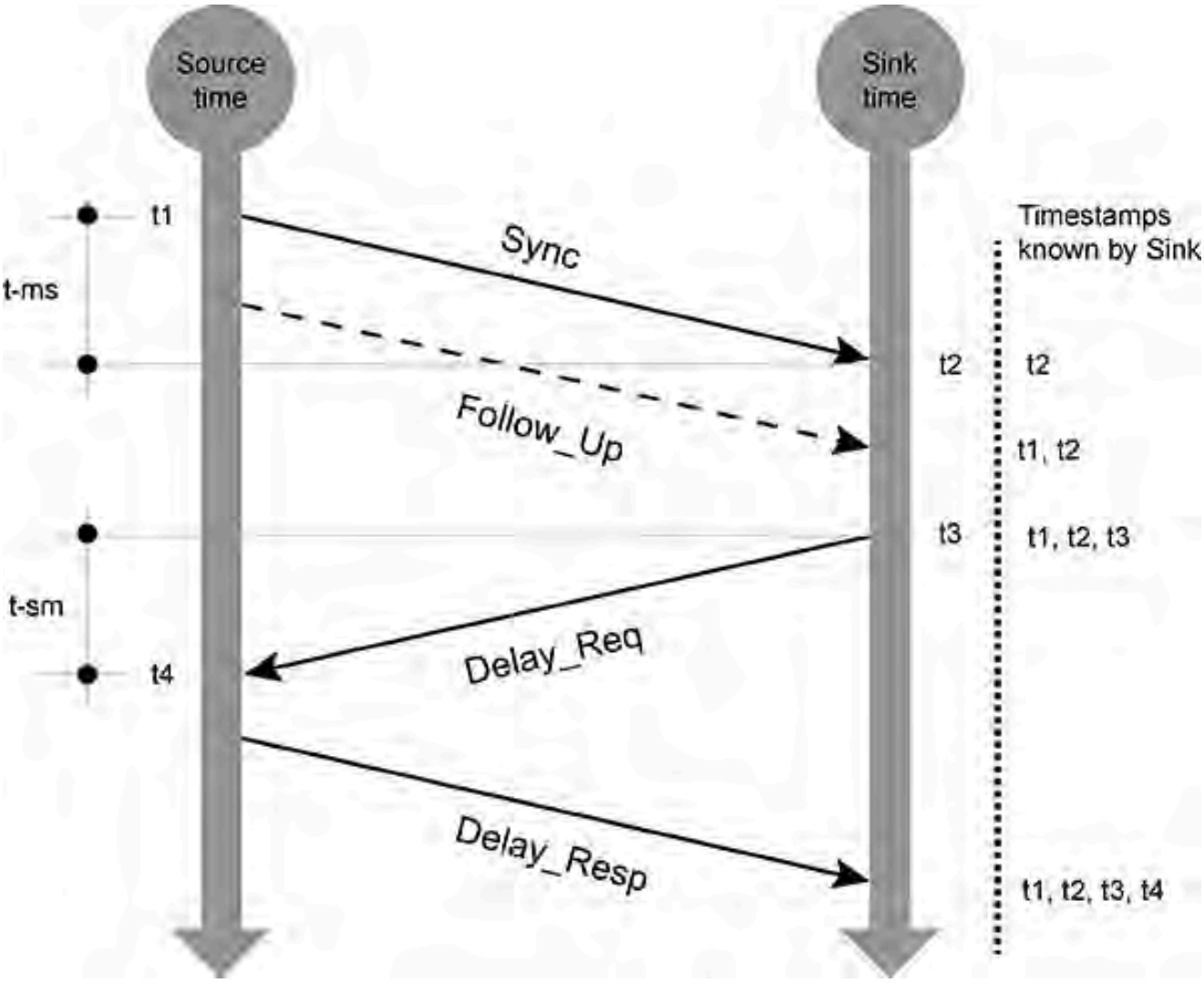
The BCM5750X controller supports two-step synchronization for synchronizing the time of day between the source clock and the sink clocks. In the two-step time synchronization process, the source clock sends a sync message to the sink clocks, and the sink clocks send back a follow-up message containing additional information about the timing of the sync message. The source clock then uses this information to compute the delay between the source clock and the sink clocks and sends a delay response message back to the sink clocks. The sink clocks use this information to further adjust their internal clocks and improve the accuracy of the time synchronization. The following figure illustrates how the two-step synchronization operates.

The general formula for two-step synchronization is as follows:

Delay = $(t2 – t1) – ½ (t_{ms} + t_{sm})$

= $(t2 – t1) – ½\{(t4 - t1) – (t3 – t2)\}$

Figure 24: Two-Step Synchronization



Installing PTP

Provides information on installing PTP.

In Linux, PTP support in the software stack is provided by a package known as linuxptp, a PTPv2 implementation according to the IEEE standard 1588 for Linux. The linuxptp package includes the ptp4l and phc2sys programs for clock synchronization. The ptp4l program implements the PTP boundary clock and ordinary clock. Hardware time stamping is used to synchronize the PTP hardware clock to the source clock. With software time stamping, it synchronizes the system clock to the source clock. The phc2sys program is needed only with hardware time stamping for synchronizing the system clock to the PTP hardware clock on the network interface card (NIC).

To install linuxptp in Centos, use `yum install linuxptp` . To install linuxptp in Ubuntu, use `apt-get install linuxptp` . This installs ptp4l and phc2sys.

Checking for Driver and Hardware Support

To use PTP, the kernel network driver for the intended interface has to support either software or hardware time stamping capabilities. In addition to hardware time stamping support being present in the driver, the Ethernet adapter, in this case the BCM5750X, must also be capable of supporting this functionality in the physical hardware. The best way to verify the time stamping capabilities of a particular driver and Ethernet adapter is to use the ethtool utility to query the interface as follows:

```
[root@localhost ~]# ethtool -T ens106f0
Time stamping parameters for ens106f0:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 3
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
        ptpv2-l4-event        (HWTSTAMP_FILTER_PTP_V2_L4_EVENT)
        ptpv2-l2-event        (HWTSTAMP_FILTER_PTP_V2_L2_EVENT)
```

For software time stamping support, the parameters list should include the following:

```
SOF_TIMESTAMPING_SOFTWARE
SOF_TIMESTAMPING_TX_SOFTWARE
SOF_TIMESTAMPING_RX_SOFTWARE
```

For hardware time stamping support, the parameters list should include the following:

```
SOF_TIMESTAMPING_RAW_HARDWARE
SOF_TIMESTAMPING_TX_HARDWARE
SOF_TIMESTAMPING_RX_HARDWARE
```

As seen from the previous figure, the BCM5750X Ethernet adapter supports hardware time stamping and is capable of providing hardware timestamps for both transmit and receive packets with different filter modes.

Configuring PTP

Provides information on configuring PTP.

In this section, two BCM5750X Ethernet adapters are configured with PTP, connected back-to-back, with one acting as source and the other as sink.

For ptp4l, the command line options and other options, that cannot be set on the command line, can be set in an optional configuration file. The configuration file must be specified at runtime with the -f option.

The configuration files for the profiles discussed in PTP Profiles can be found under directory `/usr/share/doc/linuxptp/configs/` when linuxptp is installed using yum.

In the ptp4l conf file, add the parameter `tx_timestamp_timeout` with a value of 50 to allow slightly more time for the driver's work thread to fetch the TX timestamp from the chip.

On the source, run the below command:

```
ptp4l -i <NIC interface name> -m -4 -f /etc/ptp4l.cfg
```

On the sink, run the below command:

```
ptp4l -i <NIC interface name> -m -s -4 -f /etc/ptp4l.cfg
```

The client will be in sync with the source as seen in the following figure where the sink is in S2 servo state with minimal changes to the freq and delay offsets.

Refer to ptp4l Configuration File for the contents of the ptp4l config file used in the previous instructions.

```
[root@dhcp-10-193-55-197 ~]# ptp4l -i ens106f0 -m -s -4 -f /etc/ptp4l.cfg
ptp4l[179585.569]: selected /dev/ptp2 as PTP clock
ptp4l[179585.612]: port 1: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[179585.613]: port 0: INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[179590.324]: port 1: new foreign master b02628.fffe.f55d1c-1
ptp4l[179592.655]: selected local clock bc97e1.fffe.ab6d78 as best master
ptp4l[179594.324]: selected best master clock b02628.fffe.f55d1c
ptp4l[179594.324]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[179596.324]: master offset   -1989300 s0 freq  -11608 path delay        59
ptp4l[179597.324]: master offset   -1989467 s1 freq  -11775 path delay        59
ptp4l[179598.324]: master offset       -829 s2 freq  -12604 path delay        59
ptp4l[179598.324]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[179599.324]: master offset          3 s2 freq  -12021 path delay        59
ptp4l[179600.324]: master offset        290 s2 freq  -11733 path delay        15
ptp4l[179601.324]: master offset        249 s2 freq  -11687 path delay        15
ptp4l[179602.324]: master offset        157 s2 freq  -11704 path delay        19
ptp4l[179603.324]: master offset         86 s2 freq  -11728 path delay        19
ptp4l[179604.324]: master offset         38 s2 freq  -11750 path delay        19
ptp4l[179605.324]: master offset         17 s2 freq  -11760 path delay        17
ptp4l[179606.324]: master offset          2 s2 freq  -11770 path delay        16
ptp4l[179607.324]: master offset         -2 s2 freq  -11773 path delay        14
ptp4l[179608.324]: master offset         -2 s2 freq  -11774 path delay        13
ptp4l[179609.325]: master offset         -2 s2 freq  -11774 path delay        13
ptp4l[179610.325]: master offset         -1 s2 freq  -11774 path delay        13
ptp4l[179611.324]: master offset          0 s2 freq  -11773 path delay        11
ptp4l[179612.325]: master offset          3 s2 freq  -11770 path delay         9
ptp4l[179613.325]: master offset         -2 s2 freq  -11774 path delay        10
ptp4l[179614.325]: master offset         -1 s2 freq  -11774 path delay        10
ptp4l[179615.326]: master offset          2 s2 freq  -11771 path delay         8
ptp4l[179616.325]: master offset          1 s2 freq  -11772 path delay         8
ptp4l[179617.325]: master offset          1 s2 freq  -11771 path delay         8
```

Synchronize System Times of Source and Sink

Start phc2sys to synchronize the system times between the source and the sink.

On source, run the following command:

```
phc2sys -s CLOCK_REALTIME -c <NIC interface name> -m –w
```

On sink, run the following command:

```
phc2sys -s <NIC interface name> -c CLOCK_REALTIME -m –w
```

After several seconds, the system clocks are synchronized.

Ensure that phc2sys on the source and client are not reporting large maxed-out frequency corrections. The following figure shows an acceptable phc offset with respect to CLOCK_REALTIME.

```
phc2sys[179711.498]: CLOCK_REALTIME phc offset    813289 s0 freq   +5251 delay    763
phc2sys[179712.499]: CLOCK_REALTIME phc offset    503961 s1 freq -303928 delay    732
phc2sys[179713.499]: CLOCK_REALTIME phc offset    142573 s2 freq -161355 delay    729
phc2sys[179714.500]: CLOCK_REALTIME phc offset    240842 s2 freq  -20314 delay    758
phc2sys[179715.500]: CLOCK_REALTIME phc offset    248632 s2 freq  +59728 delay    769
phc2sys[179716.500]: CLOCK_REALTIME phc offset    189917 s2 freq  +75603 delay    770
phc2sys[179717.500]: CLOCK_REALTIME phc offset    112714 s2 freq  +55375 delay    732
phc2sys[179718.500]: CLOCK_REALTIME phc offset     54269 s2 freq  +30744 delay    768
phc2sys[179719.500]: CLOCK_REALTIME phc offset     24912 s2 freq  +17668 delay    769
phc2sys[179720.501]: CLOCK_REALTIME phc offset     13923 s2 freq  +14153 delay    760
phc2sys[179721.501]: CLOCK_REALTIME phc offset      7117 s2 freq  +11523 delay    737
phc2sys[179722.501]: CLOCK_REALTIME phc offset      2714 s2 freq   +9256 delay    760
phc2sys[179723.501]: CLOCK_REALTIME phc offset       -32 s2 freq   +7324 delay    766
phc2sys[179724.501]: CLOCK_REALTIME phc offset     -1668 s2 freq   +5678 delay    763
phc2sys[179725.502]: CLOCK_REALTIME phc offset     -1745 s2 freq   +5101 delay    763
phc2sys[179726.502]: CLOCK_REALTIME phc offset     -1739 s2 freq   +4583 delay    759
phc2sys[179727.502]: CLOCK_REALTIME phc offset      -925 s2 freq   +4876 delay    759
phc2sys[179728.502]: CLOCK_REALTIME phc offset      -305 s2 freq   +5218 delay    763
phc2sys[179729.502]: CLOCK_REALTIME phc offset        -9 s2 freq   +5423 delay    763
phc2sys[179730.502]: CLOCK_REALTIME phc offset        70 s2 freq   +5499 delay    758
phc2sys[179731.503]: CLOCK_REALTIME phc offset        53 s2 freq   +5503 delay    763
phc2sys[179732.503]: CLOCK_REALTIME phc offset        81 s2 freq   +5547 delay    766
phc2sys[179733.503]: CLOCK_REALTIME phc offset        29 s2 freq   +5519 delay    766
phc2sys[179734.503]: CLOCK_REALTIME phc offset       -45 s2 freq   +5454 delay    759
phc2sys[179735.503]: CLOCK_REALTIME phc offset       -44 s2 freq   +5441 delay    763
phc2sys[179736.504]: CLOCK_REALTIME phc offset       -20 s2 freq   +5452 delay    763
phc2sys[179737.504]: CLOCK_REALTIME phc offset        -8 s2 freq   +5458 delay    767
phc2sys[179738.504]: CLOCK_REALTIME phc offset       -31 s2 freq   +5433 delay    772
phc2sys[179739.504]: CLOCK_REALTIME phc offset       -39 s2 freq   +5415 delay    768
phc2sys[179740.504]: CLOCK_REALTIME phc offset       -22 s2 freq   +5421 delay    763
phc2sys[179741.504]: CLOCK_REALTIME phc offset       -14 s2 freq   +5422 delay    768
phc2sys[179742.505]: CLOCK_REALTIME phc offset         2 s2 freq   +5434 delay    758
phc2sys[179743.505]: CLOCK_REALTIME phc offset         1 s2 freq   +5433 delay    760
phc2sys[179744.505]: CLOCK_REALTIME phc offset        14 s2 freq   +5447 delay    768
phc2sys[179745.505]: CLOCK_REALTIME phc offset         7 s2 freq   +5444 delay    769
phc2sys[179746.505]: CLOCK_REALTIME phc offset         0 s2 freq   +5439 delay    766
phc2sys[179747.505]: CLOCK_REALTIME phc offset        -2 s2 freq   +5437 delay    768
```

All clocks are now synchronized.

For best results use the following order:

1. Start phc2sys on both source and sink (they will not start and will show a message waiting for ptp4l).
2. Start ptp4l on source and then on sink.
3. The source is in Grandmaster mode. phc2sys on the source starts, sink ptp4l goes into s2 state, and finally phc2sys on the sink starts.

This ensures that when ptp4l starts, if there are large adjustments to be made, it is taken care of in s0/s1 state before entering s2 state. Once ptp4l is in s2 state, only small adjustments can be made.

ptp4l Configuration File

Provides the ptp4l configuration file.

The following is the ptp4l config file that is used in the above example.

```
cat /etc/ptp4l.cfg
[global]
tx_timestamp_timeout 50
```

Note: The software now reports `tx_pkts` , `tx_lost` , and `tx_err` when timestamping is enabled. This can be helpful when determining if the configuration was done correctly.

Tuning Ethernet Network Adapters

Provides information on fine-tuning Ethernet network adapters for improved performance.

Note: Bnxtnvm commands are no longer supported in the 228 version of the User Guide. Bnxtnvm will be replaced with NICCLI beginning with the 230 release. NICCLI has been architected from the ground up with security considerations. Refer to the NICCLI Commands moving forward. If bnxtnvm commands are required, refer to the 227 version of the User Guide (EtherNIC-Ctrl-UG2-27) located in ESP (esp.broadcom.com). Contact your Broadcom representative for additional questions.

This section provides the following tuning information for the Broadcom Ethernet network adapter:

- Automatic Tuning for Linux Systems
- BIOS Tuning on Ethernet Network Adapters
- TCP Performance Tuning on Ethernet Network Adapters
- DPDK Tuning for Ethernet Network Adapters
- IP Forwarding Tunings for Ethernet Network Adapters
- RoCE Tuning for Ethernet Network Adapters

NIC Tune

Provides information on automatic tuning for Linux systems using the NIC Tune utility.

Quickstart

The `nictune.py` tool reports status on a list of tunable parameters of all supported network interface cards when invoked as:

```
nictune.py
```

To optimally tune all supported network adapters for the default maximum throughput profile, and install scripts to persist those changes between reboots, invoke as:

```
nictune.py -t -i
```

See `nictune.py --help` output for all available options.

Introduction

This tool automates performance tuning for Broadcom Ethernet Network Interface Cards (NICs). The following tuning parameters are managed:

- NIC configured performance profile
- Ethtool TX/RX/combined queue sizes
- Hardware generic receive offload (GRO)
- Interrupt CPU affinity (IRQ Affinity)
- Kernel IOMMU
- Kernel performance governor
- TCP buffer memory
- Transmit packet steering (XPS)
- Receive Flow Steering (RFS)
- Interrupt Coalescing

The following parameters are managed on RoCE profile:

- NIC PCIe Relaxed Ordering enablement
- NIC User Defined Congestion Control
- RoCE DCSP TLV configured
- CNP DCSP TLV configured
- CoS Queue PFC enabled Lossless

The tool has the following modes:

- Report: Without arguments, the tool reports status of each tuning parameter for each supported NIC in the system.
- Tune: The `-t` argument applies optimal values to the running system, but these are reset after any reboot.
- Install: The `-i` argument installs boot-time scripts to configure the tuning parameters, making the change permanent.
- Uninstall: The `-u` argument removes the installed scripts from any previous run with the `-i` argument.
- Performance Benchmarking with iperf3: The `-P` argument provides support for iperf3 performance benchmarking.

Profiles

The tool supports two types of profiles for the different parameters when invoked by the `-p` argument:

- Throughput: A default profile that provides the parameters that are focused on L2 performance tuning.
- RoCE: `-p roce` provides parameters that are related to RoCE features and only supports reporting mode.

Performance Test Mode

The tool supports performance testing by `iperf` or `iperf3` . The`--runperf` argument runs the performance test.

- Client mode: The `--perfclient` argument runs the performance client, connecting to the specified IP address.
- Server mode: The `--perfserver` argument runs the performance server process.

Note: The `-P` or `--parallel` option is available to enable concurrent processing when using the `iperf3` tool. This option has no effect when used with the `iperf` tool.

Multiple IP addresses can be assigned with multiple ports at the same time, or you can select a specific device with `-d` .

The result will show at the end of output for each devices, as shown below:

```
[RESULT] Performance result on eth0: xxx Gbit/s
[RESULT] Performance result on eth1: xxx Gbit/s
```

Examples

System-A and System-B have installed N2200GDP (2x200G) adapters and the interface names are eth0 and eth1.

```
In system-A, the IP address of eth0 is 192.168.1.1 and the IP address of eth1 is 192.168.2.1
In system-B, the IP address of eth0 is 192.168.1.2 and the IP address of eth1 is 192.168.2.2
```

Case 1: Run uni-directional traffic on both ports between System-A and System-B after applying optimal tuning values `(-t)` .

```
System-A with server mode: ./nictune.py -t --runperf --perfserver
System-B with client mode: ./nictune.py -t --runperf --perfclient 192.168.1.1,192.168.2.1
```

Case 2: Run bi-directional traffic on both ports between System-A and System-B after applying optimal tuning values `(-t)` .

```
System-A : ./nictune.py -t --runperf --perfserver --perfclient 192.168.1.2,192.168.2.2
System-B : ./nictune.py -t --runperf --perfserver --perfclient 192.168.1.1,192.168.2.1
```

Case 3: Run bi-directional traffic on eth0 only between System-A and System-B after applying optimal tuning values `(-t)` .

```
System-A : ./nictune.py -t -d eth0 --runperf --perfserver --perfclient 192.168.1.2
System-B : ./nictune.py -t -d eth0 --runperf --perfserver --perfclient 192.168.1.1
```

Requirements

Supported Operating Systems

See the Supported Operating Systems for Ethernet Network Adapters page for the full list of supported operating systems and platforms.

Supported Devices

The following Broadcom devices are fully supported by this tool:

- BCM95741X
- BCM95750X
- BCM957608

All speeds and cable/transceiver types supported by the above devices are supported by this tool.

Software Requirements

The Broadcom driver (`bnxt_en`) must be installed and configured. The Broadcom NICCLI tool must also be installed to manage some tuning parameters. The `iperf/iperf3` utility must be installed to run the performance testing. See Installing Software for Ethernet Network Adapters for additional information.

Performance Profiles

Provides information on choosing a performance profile based on the workload.

Currently, two profiles are supported:

- Default Profile – optimized for non-RoCE (L2/IP) performance
- RoCE Profile – optimized for RoCE performance

NICCLI Commands

To set the default profile, use the following command:

```
niccli -i <index> nvm --setoption performance_profile --value 0
```

To set the RoCE profile, use the following command:

```
niccli -i <index> nvm --setoption performance_profile --value 1
```

To query the current profile, use the following command:

```
niccli -i <index> nvm --getoption performance_profile
```

Note: It is recommended to use the RoCE profile for RoCE traffic and default profile for non-RoCE traffic.

BIOS Tuning on Ethernet Network Adapters

Provides the BIOS configuration options to tune the system for optimal performance.

The BIOS screens in this section are for reference only and have been captured on the AMD EPYC reference platform. It is recommended to find the equivalent settings in the target system BIOS console.

This section provides the following BIOS tuning information for the Broadcom Ethernet network adapter:

- NPS (NUMA Per Socket)
- X2APIC
- Determinism Control and Determinism Slider
- APBDIS
- Preferred I/O and Enhanced Preferred I/O
- PCIe Ten Bit Tag
- Memory Clock Speed
- L3 LLC (Last Level Cache) as NUMA
- Socket/Inter-Chip Global Memory Interconnect (xGMI)

The following section provides BIOS settings for different CPU models:

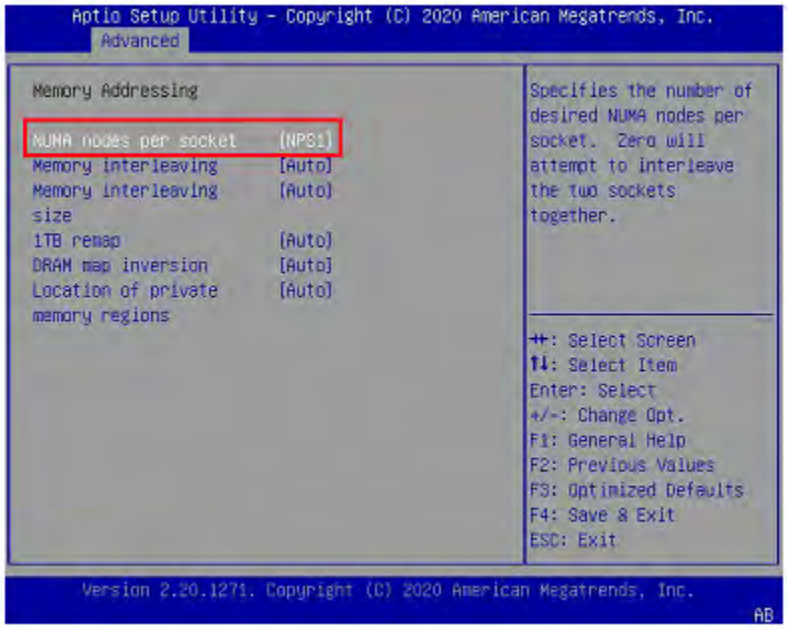
- TCP BIOS Performance Tuning

NPS (NUMA Per Socket)

Provides information on configuring the number of NPS (NUMA per socket) nodes.

To access the NUMA nodes per socket setting, select **Advanced** > **AMD CBS** > **DF Common Options** > **Memory Addressing** > **NUMA Nodes Per Socket** > **NPS1 Socket** > **NPS1**.

Figure 25: NUMA Nodes Per Socket Settings



Note: Use NPS[1] configuration for 200 Gb/s and above. Use [NPS4] for speeds up to 100 Gb/s, which provides better CPU and memory locality. If the BIOS includes memory interleaving as a distinct option, it is recommended to leave it in either the Auto or Enable state.

X2APIC

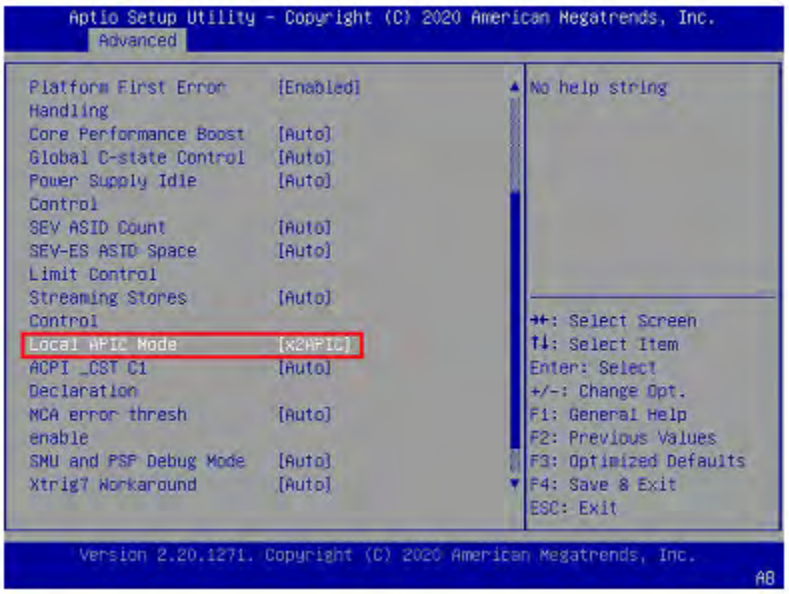
Provides information on enabling X2APIC for improved performance.

Set the local APIC mode to **x2APIC = Enabled** to allow the operating system to work with 256 threads and improve performance over legacy APIC.

Note: Disable SMT if you are running an operating system that does not support X2APIC and has a dual-socket 64 core processor.

To access the **Local APIC Mode** setting, select **Advanced** > **AMD CBS** > **CPU Common Options** > **Local APIC Mode** > **X2APIC** > **NPS1 Socket**

Figure 26: Local APIC Mode Settings



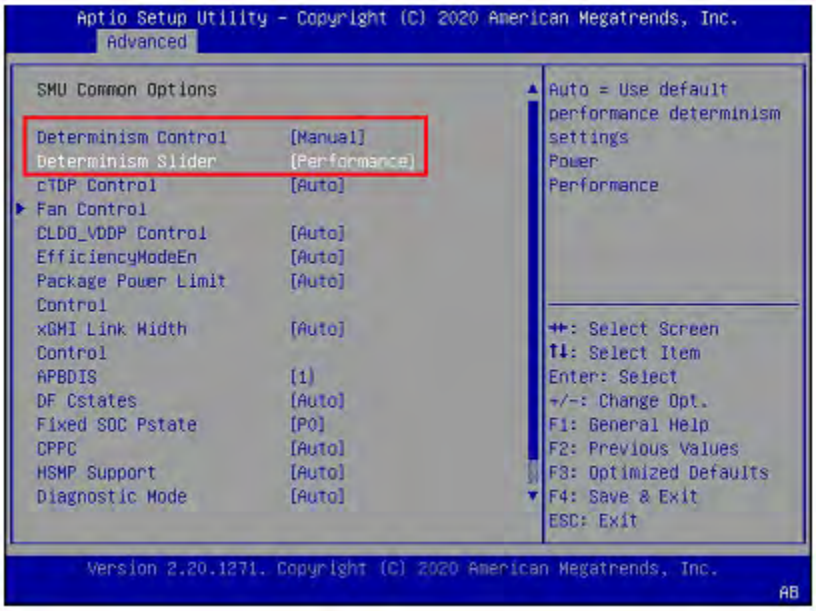
Determinism Control and Determinism Slider

Provides information on setting determinism control and the determinism slider for consistent performance.

Set **Determinism Control** to **Manual** and the **Determinism Slider** to **Performance** (see the following figure) to ensure consistent performance across a fleet of similarly configured systems.

1. To access the **Determinism Control** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **SMU Common Options** > **Determinism Control** > **Manual**.
2. To access the **Determinism Slider** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **Determinism Slider** > **Performance**.

Figure 27: Determinism Control/Determinism Slider Settings



APBDIS

Provides information on disabling APBDIS.

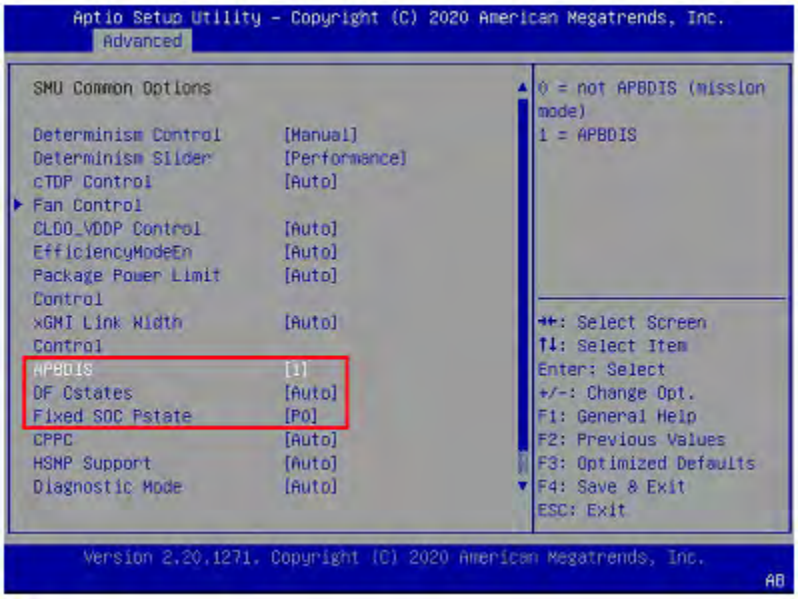
Set **APBDIS=1** to disable Algorithmic Performance Boost which subsequently disables the switching of P- states in infinity fabric (CPU P-states remain unaffected) and forces the system to be in P0 state, which is the highest performing infinity fabric P-state. The APBDIS states are as follows:

- 0: Disable APBDIS – Locks the fabric clock to the non-boosted speeds.
- 1: Enable APBDIS – Unlocks the fabric clock to the boosted speeds.
- Auto (Default setting) – Use the default value for APBDIS. The default value is 0.

To access the **APBDIS** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **SMU Common Options** > **APBDIS** > **1**.

To access the **Fixed SOC Pstate** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **SMU Common Options** > **Fixed SOC Pstate** > **P0**.

Figure 28: APBDIS Settings



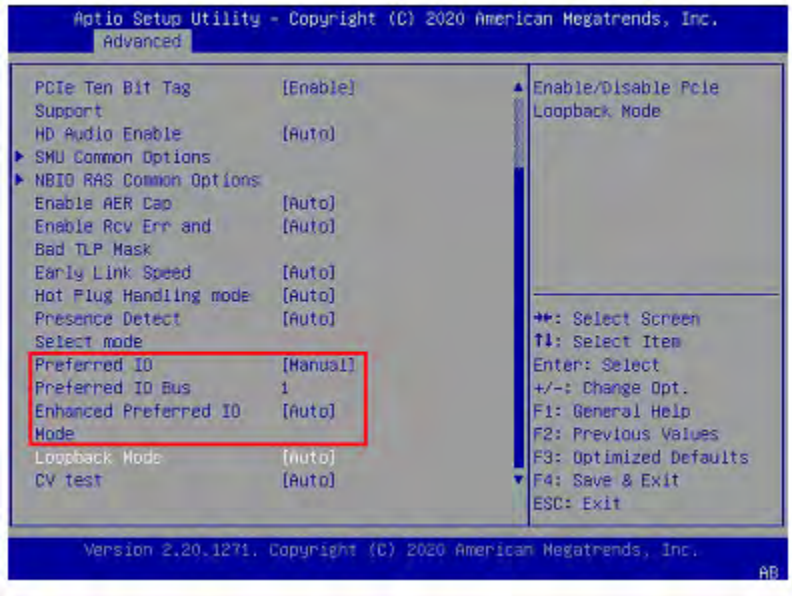
Preferred I/O and Enhanced Preferred I/O

Provides information on setting the preferred I/O and enhanced preferred I/O for improved performance.

Preferred I/O is a new capability in the EPYC 7002 series BIOS that prioritizes the traffic from the selected I/O device and facilitates the ordering of PCIe packets which reduces the overhead and results in better adapter performance.

The Enhanced Preferred I/O capability further ensures that the same configured I/O device remains at the highest performance by keeping its clocks at the maximum frequency.

1. To access the **Preferred IO** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **Preferred I/O** > **Manual**.
2. To access the **Preferred IO Bus** setting select **Advanced** > **AMD CBS** > **NBIO Common Options** > **Preferred I/O Bus** > **[PCIe Bus Number]**.
3. To access the **Enhanced Preferred IO** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **Enhanced Preferred I/O Mode** > **Auto/Enable P0**.



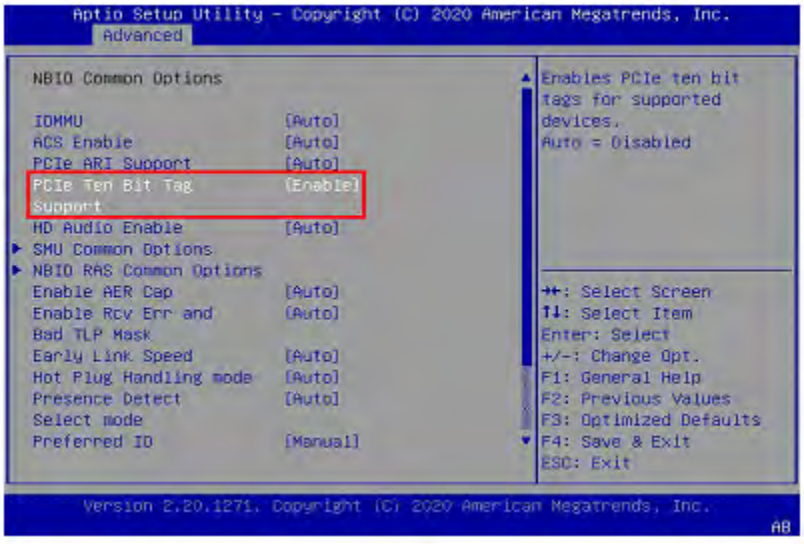
PCIe Ten Bit Tag

Provides information on enabling the PCIe ten-bit tag for improved performance.

Enable the **PCIe Ten Bit Tag** to increase the number of non-posted requests from 256 to 768 for better performance. As latency increases, an increase in unique tags is required to maintain the peak performance at 16 GT/s.

To access the **PCIe Ten Bit Tag Support** setting, select **Advanced** > **AMD CBS** > **NBIO Common Options** > **PCIe Ten Bit Tag Support** > **Enable**.

Figure 29: PCIe Ten Bit Tag Settings



Memory Clock Speed

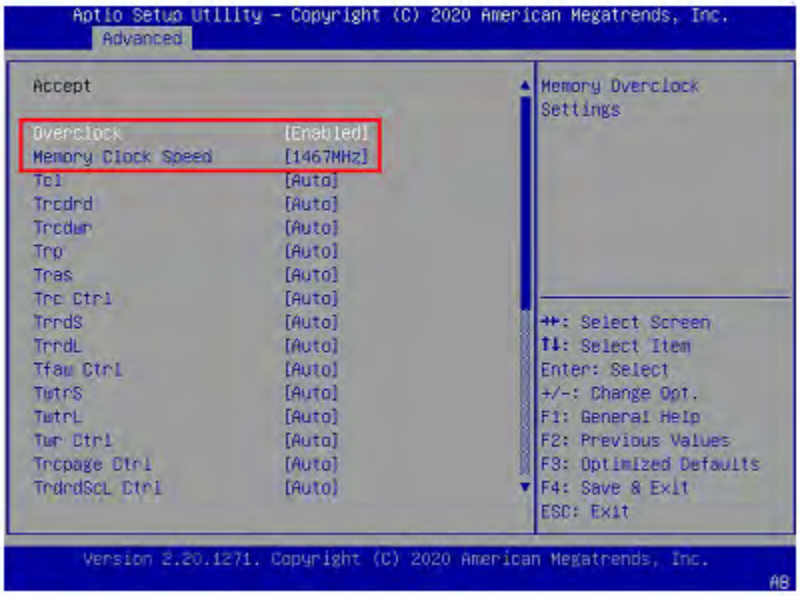
Provides information on setting the memory clock speed to match the fabric speed.

Set the **Memory Clock Speed** to match the maximum fabric clock speed supported by the installed EPYC 7002 series server, which is either 1467 MHz or 1333 MHz (the double data rate is 2x this clock – for example, MCLK = 1467 means 2933-MTS data rate).

Note: A platform that can support higher speed memory (for example, 1600 MHz memory clock) might increase the overall platform memory bandwidth; however, the average memory latency is higher.

1. To access the **Overclock** setting, select **Advanced** > **AMD CBS** > **UMC Common Options** > **DDR4 Common Options** > **DRAM Timing Configuration** > **Accept** > **Overclock** > **Enabled**.
2. To access the **Memory Clock** setting, select **Advanced** > **AMD CBS** > **UMC Common Options** > **DDR4 Common Options** > **DRAM Timing Configuration** > **Accept** > **Memory Clock Speed** > **1467MHz**.

Figure 30: Memory Clock Speed Settings



L3 LLC (Last Level Cache) as NUMA

Provides information on enabling L3 LLC (last level cache) to create NUMA nodes.

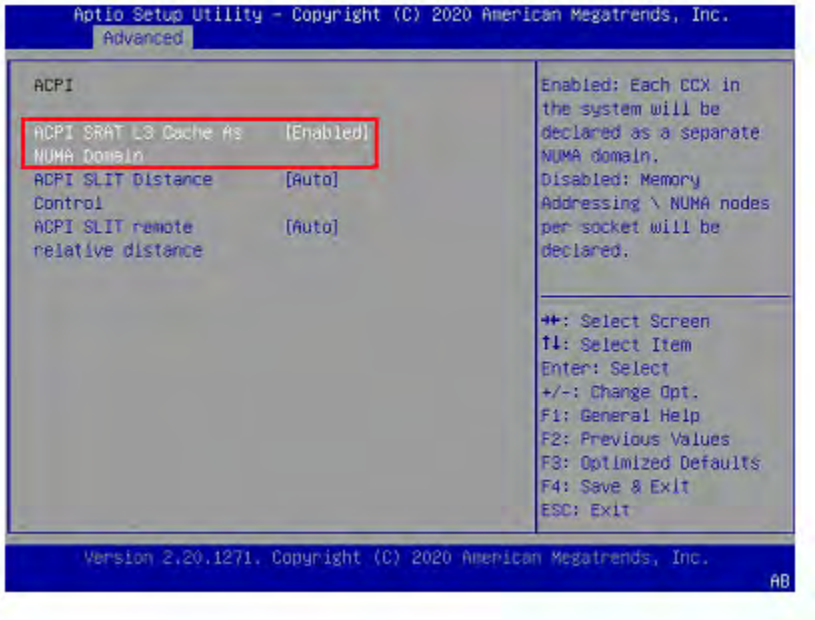
Enable L3 as NUMA to create NUMA nodes equal to the number of L3 Caches (CCX). This helps the operating system schedulers maintain locality to the last level cache (LLC) without causing unnecessary cache-to-cache transactions and improves the performance.

Note: This benchmarking feature is meant for isolating L3 caches and is not recommended for production deployments.

To access the **ACPI** settings, select **Advanced** > **AMD CBS** > **DF Common Options** > **ACPI** > **ACPI SRAT L3 cache As NUMA Domain** > **Enabled***.

* This option should be set to auto or disabled for EPYC 7003 and above series processors.

Figure 31: ACPI SRAT L3 cache as NUMA Domain Setting



Socket/Inter-Chip Global Memory Interconnect (xGMI)

Provides information on the socket/inter-chip global memory interconnect and its impact on latency.

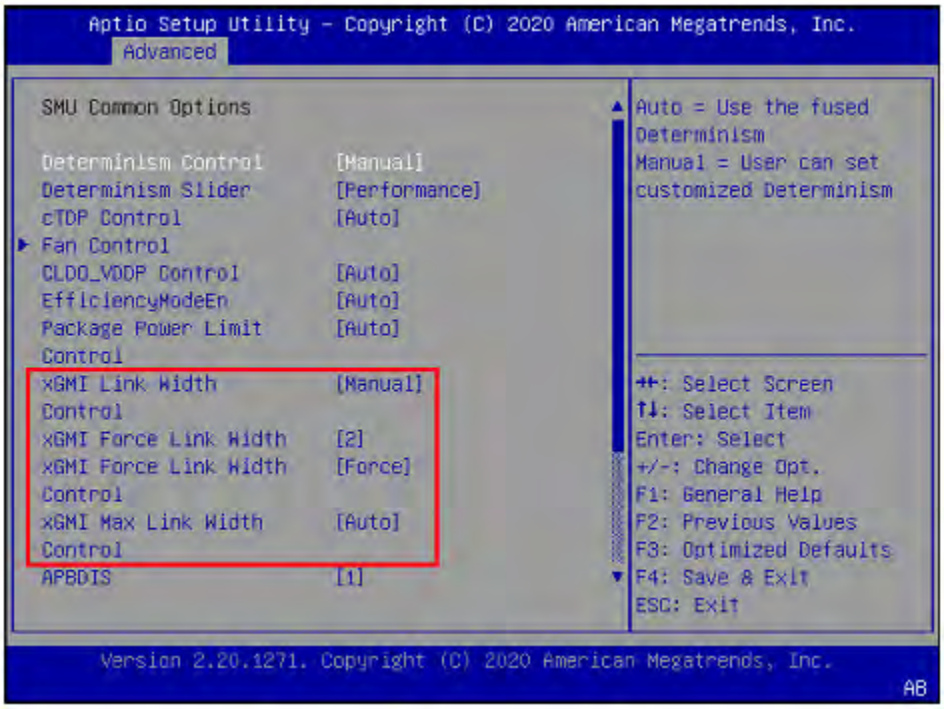
xGMI Dynamic Link Width Management saves power during periods of low socket-to-socket data traffic by reducing the number of active xGMI lanes per link from 16 to 8. However, under certain scenarios involving low bandwidth, but latency-sensitive traffic, the transition from low-power to full-power xGMI can adversely impact latency.

Setting **xGMI Link Width Control** to **Manual** and specifying a **Max Link Width** of 16 forces the xGMI interface into full-power mode, eliminating any latency jitter.

Note: The Socket/Inter-Chip Global Memory Interconnect option only applies to a 2P system.

1. To access the **xGMI Link Width Control** setting, select **Advanced** > **AMD CBS** > **SMU Common Options** > **xGMI Link Width Control** > **Manual**.
2. To access the **xGMI Force Link Width** setting, select **Advanced** > **AMD CBS** > **SMU Common Options** > **xGMI Force Link Width** > **2**.
3. To access the **xGMI Force Link Width Control** setting, select **Advanced** > **AMD CBS** > **SMU Common Options** > **xGMI Force Link Width Control** > **Force**.
4. To access the **xGMI Max Link Width Control** setting, select **Advanced** > **AMD CBS** > **SMU Common Options** > **xGMI Force Link Width Control** > **Auto**.

Figure 32: Socket/Inter-Chip Global Memory Interconnect (xGMI) Settings



Applications that are known to be insensitive to both socket-to-socket bandwidth and latency can set a fixed link width of eight to save power, which can divert more power to the cores for boost.

TCP Performance Tuning on Ethernet Network Adapters

Provides performance tuning information for TCP on Ethernet network adapters.

This section provides the following TCP performance tuning information:

- BIOS Tuning
- Adapter Tuning
- Operating System Tuning (Linux)
- TCP Example with the BCM957508-P2100G
- Configuring RSS for Performance in Windows

TCP BIOS Performance Tuning

Provides information on setting BIOS options for improved performance.

See BIOS Tuning and set the following BIOS options:

Intel Skylake (Gen3)

- HyperThreading/Logical Cores – Disable
- System Profile – Performance

AMD Milan (Gen4)/AMD Genoa (Gen5)

- Logical Processor – Disabled
- IOMMU – Enabled
- NPS – 1
- L3 Cache as NUMA – Disabled
- X2APIC Mode – Enabled
- PCIe Ten Bit Tag – Enabled
- Preferred IO Bus – Enabled (Milan Only)
- Preferred IO Bus Value – Provide BUS ID (Milan Only)
- Enhanced Preferred IO – Enabled (Milan Only)
- CPU Power Management – Maximum Performance
- Memory Frequency – Maximum Performance
- Determinism Slider – Performance
- APBDIS – Enabled
- xGMI Link Width Control – Manual
- xGMI Force Link Width – 2
- xGMI Force Link Width Control – Force
- xGMI Max Link Width Control – Auto

Intel Icelake (Gen4)/Intel Sapphire Rapids (Gen5)

- Logical Processor – Disabled
- Virtualization Technology – Enabled
- Sub NUMA Cluster – Disabled
- MADT Core Enumeration – Round Robin
- I/O Snoop HoldOff Response – 2K Cycles
- SR-IOV Global Enable – Disabled
- System Profile – Performance
- x2APIC Mode – Enabled

Adapter Tuning

Provides information on setting adapter options for improved performance.

Note: Contact the OEM to get the latest driver, firmware, and tools, and follow the installation instructions.

This section provides the following adapter tuning information for the Broadcom Ethernet network adapters:

- Performance Profiles
- NUMA: Local vs. Non Local
- Configuring Queues
- Configuring IRQ and Application Affinity
- TX and RX Flow Steering
- TX and RX Queue Size
- Interrupt Moderation
- GRO (Generic Receive Offload)
- Relaxed Ordering
- PCIe MRRS (Maximum Read Request Size)

NUMA: Local vs. Non Local

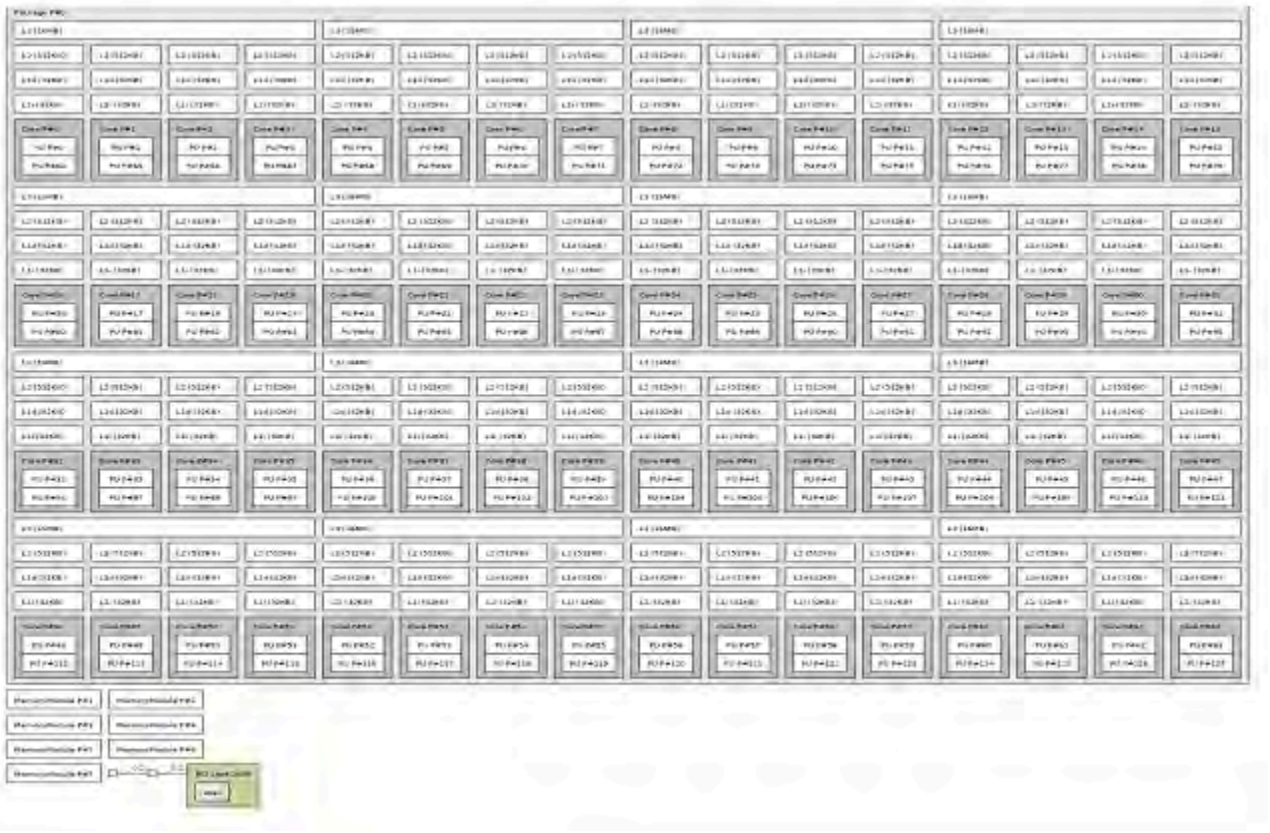
Provides information using NUMA for faster memory access.

Non Uniform Memory Access (NUMA) is a memory architecture in which each CPU is connected to its local memory. The local NUMA CPUs provide faster access to the local memory (shorter distance), accessing the memory on remote NUMA is possible, but it is slower.

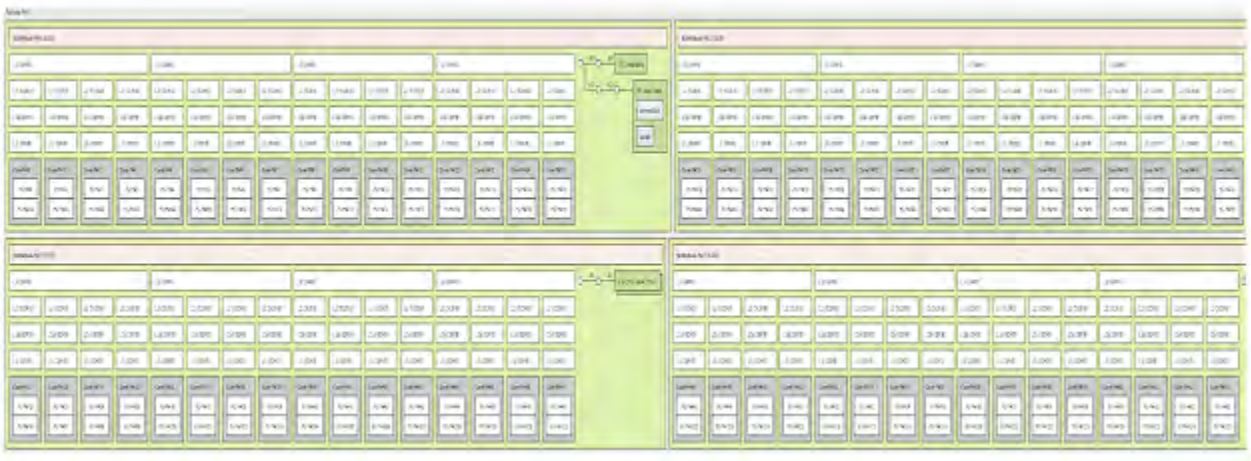
- Local NUMA: `cat /sys/class/net/[interface]/device/numa_node`
- Local CPUs: `cat /sys/class/net/[interface]/device/local_cpulist`

Alternatively, install *hwloc* and use the output of *lstopo* to find the local NUMA nodes and cpulist:

```
yum install hwloc hwloc-gui
lstopo --logical --output-format png > cpu.png
```



Note: Single Socket NPS=1, all cores appear local.



Note: Single Socket NPS=4, only a single node is local.

For previous software releases (2.28 and earlier), the Page Pool (a Linux kernel networking stack construct used for allocating memory to receive packets) looked at the node local to the PCIe device to determine where to allocate memory. For example, a host system has 2 NUMA nodes -- node 0 with 'n' cores and node 1 with 'm' cores. If a PCIe device was on NUMA node 0 , then all packet receiver buffer memory allocations would be on Node 0 even if the number of rings of the device exceeds the number of cores on that NUMA node 0 (For example, the Number of Rings on the device is more than n). Starting with software version 2.29, this behavior has changed such that RX ring packet buffer memory allocations are on Node 0 until resources are exhausted (the first $n rings on node 0) and then packet buffer memory allocations are accomplished on Node 1 ($n -> m - 1). This ensures the best performance for applications running on Node 1.

Configuring Queues

Provides information on combining queues using the ethtool.

Broadcom Ethernet network adapters support both combined and separate queue configuration using ethtool. For most cases, the use of combined queues is recommended. With combined queues, TX and RX queues are shared with a single IRQ.

```
ethtool -L [interface] combined 8 tx 0 rx 0
```

For granular control packet processing, but at the cost of more resources, it is recommended to use separate queues. In this configuration, the TX and RX queues are separate and have independent IRQs giving the ability to have finer control over the IRQ affinities.

```
ethtool -L [interface] combined 0 tx 4 rx 4
```

In either case, it is recommended to have no more than a single IRQ per physical core. It may be beneficial to only allocate a single IRQ per local core in a CCD.

Configuring IRQ and Application Affinity

Provides information on configuring IRQ and application affinity for improved performance.

IRQ affinity refers to the binding of interrupts from a specific device to one or multiple logical processors. The distribution of the IRQs across different local logical cores results in improved performance from better CPU utilization.

Use the following steps for IRQ affinity configuration:

1. Disable irqbalance (to prevent the service from dynamically moving your IRQ) using the following commands:

`service irqbalance stop`

`service irqbalance disable` (to keep it persistent through reboot)

2. Identify local CPUs using the following command:

```
cat /sys/class/net/[interface]/device/local_cpulist
```

3. Identify IRQ numbers using either of the following methods:
   a. Identify IRQ numbers using the following command:

```
cat /proc/interrupts | grep [interface] | awk -F ":" '{print $1}'
```

   b. Use the following script:

The script `Interrupts.py` is intended to help find IRQ associated with each queue.

```
== Interrupts.py
#!/usr/bin/env python
import sys
import argparse
import time

class interrupts:

def__init__(self, interface, queue=None, continuous=None):
self.interrupts = {}
self.queue = queue
self.continuous = continuous
self.interface = interface
self.parse_interrupts(interface, queue)
def parse_interrupts(self, interface, queue):
# cpu_count = 0
f = open("/proc/interrupts")

for line in f.readlines():
    line = line.strip()
    # if "CPU" in line:
    # fields = line.split()
    # for i in range(0,len(fields)):
    #              if "CPU" in fields[i]:
    #                                    cpu_count += 1
    # #print line
    a = interface in line and queue and line.endswith('-' + queue)
    b = interface in line and not queue
    if a or b:
        fields = line.split()
        irq = fields[0]
        if irq not in self.interrupts.keys():
            self.interrupts[irq] = {}
        f2 = open('/proc/irq/%s/smp_affinity_list' %irq.strip(':'))
        self.interrupts[irq]['cpu'] = f2.read().strip()
        self.interrupts[irq]['queue'] = fields[-1]
        # cast string values to intergers
        #interrupt_values = map(int, fields[1:-2])
        interrupt_values = map(int, fields[1:-5])
        sum_val = sum(interrupt_values)
        if 'irq_diff' not in self.interrupts[irq].keys():
                self.interrupts[irq]['irq_diff'] = 0
```

```
        else:
                self.interrupts[irq]['irq_diff'] = sum_val - self.interrupts[irq]['irq_val']
        self.interrupts[irq]['irq_val'] = sum_val
    f.close()

def print_interrupts(self):
 for key in sorted(self.interrupts.keys()):
     print"%s\t%-*s[%s] %-*s %s" % (key, 10, self.interrupts[key]['irq_val'] \
                                            , self.interrupts[key]['irq_diff'] \
                                       , 10, self.interrupts[key]['queue'],self.interrupts[key]['cpu'])
    print"==================================================================="

if __name__ == "__main__":

parser = argparse.ArgumentParser(description='Observe interrupts based on interface or queue')
parser.add_argument('interface', type=str, \
                    help='an interface')
parser.add_argument('queue', default=None, nargs='?',
                    help='the queue # associated with interface to monitor (default: All)')
parser.add_argument('-c', '--c', metavar='sleep', type=int, nargs='?', const=5, \
                    help='Run Continuously with wait between (default: 5)')

args = parser.parse_args()
print args

# if len(sys.argv) >= 2:
 #        print 'interrupts.py [interface]'
 #        sys.exit(2)
 # interface = sys.argv[1]
 ##port = sys.argv[2]
 a = interrupts(args.interface, args.queue, args.c)
 a.print_interrupts()

 while args.c:
     time.sleep(args.c)
     a.parse_interrupts(a.interface, a.queue)
     a.print_interrupts()
 ==
```

1. Find the IRQs associated with each queue using the following command:

   ```
   python interrupts.py <interface> -c
   ```

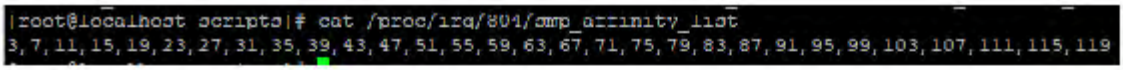
   Note: <interface> is the Linux interface of the adapter (for example, p9p1).

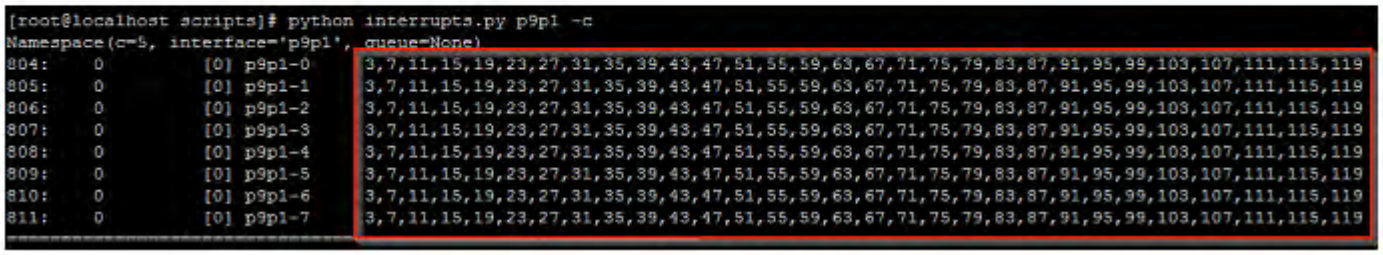
```
[root@localhost scripts]# python interrupts.py p9p1 -c
Namespace(c=5, interface='p9p1', queue=None)
304:    0       [0] p9p1-0      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
305:    0       [0] p9p1-1      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
306:    0       [0] p9p1-2      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
307:    0       [0] p9p1-3      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
308:    0       [0] p9p1-4      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
309:    0       [0] p9p1-5      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
310:    0       [0] p9p1-6      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
311:    0       [0] p9p1-7      3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79,83,87,91,95,99,103,107,111,115,119
```

   The IRQs are listed in the left-hand column.

2. Check the IRQ/queue affinity (listed in the right-hand column of `interrupts.py` output shown in the previous figure) with the following command:

   ```
   cat /proc/irq/<IRQ #>/smp_affinity_list
   ```





   By default, the IRQs are set to any processor on the same PCIe NUMA node.

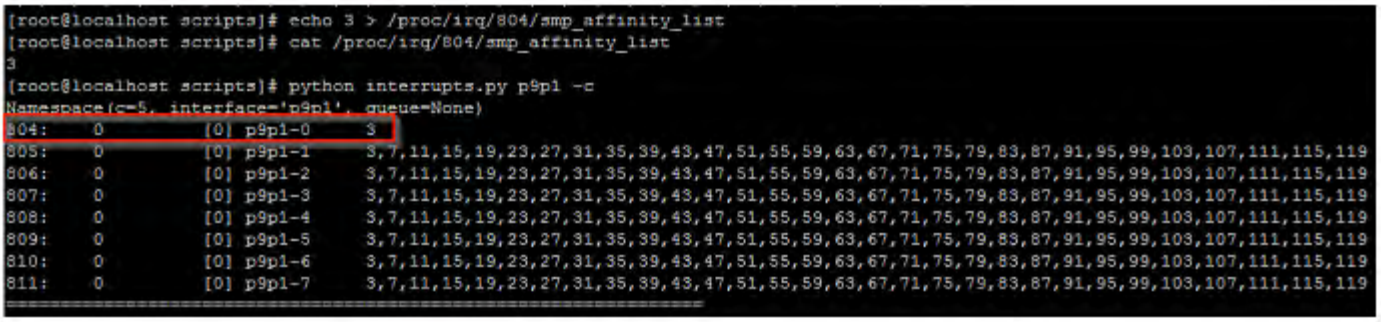
3. Set the IRQ affinity with the following command:

   ```
   echo <processor #> > /proc/irq/<IRQ #>/smp_affinity_list
   ```

   Example: `echo 3 > /proc/irq/804/smp_affinity_list`

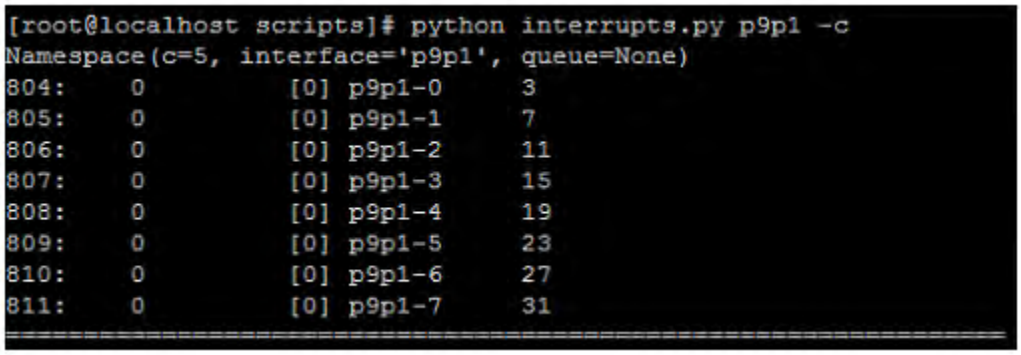
4. Check the IRQ/queue affinity (listed in the right-hand column of `interrupts.py` output below) with the following command:

   ```
   cat /proc/irq/<IRQ #>/smp_affinity_list
   ```



   It is necessary to recheck after assigning because the processor affinity (or process assignment) might not be set on certain processors.

5. Configure each queue to be assigned to a single unique processor. Make a note of these processors so that the assigned CPUs for netperf/netserver are not reused.



It is preferred to use the same CPUs for application affinity which also allows cache locality between interrupts and application threads and reduces the processing overhead. taskset and numactl tools or application-specific options (for example, netperf with -T) can be used for configuring application locality:

```
taskset -c [cpu_core list] application
```

or

```
numactl -C [cpu_cores list] application
```

or application-specific options, for example: If using netperf, a -T option handles both server and client application affinity.

TX and RX Flow Steering

Provides information on TX and RX flow steering for proper traffic distribution.

For the best performance, it is important to distribute traffic across multiple software queues. It distributes the traffic and allows the utilization of multiple CPU cores to handle the processing. There are several ways to distribute traffic across multiple software queues.

Enabled by default, RSS (Receive Side Scaling), provides a good mechanism for RX load distribution as it hashes different streams to separate RX queues to spread the load evenly. However, it does not consider application locality. For example, a flow could hash to queue 0 which is being processed on core 0 but the application consuming that data is running on core 64. This does not benefit from any locality. However, if the processing is highly CPU intensive there could be a benefit to having the application and IRQ on different processors. RFS (Receive Flow Steering) overcomes this shortcoming by steering the packets to the CPU cores where the application thread is running thus increasing the data cache hit rate. Further, Broadcom NICs support this steering in HW, aRFS (Accelerated RFS).

The TX flow steering can be achieved by configuring XPS (Transmit Packet Steering) which guarantees that TX Interrupts are generated on the same core running the application.

Configure XPS by setting the appropriate local CPU mask for every TX queue as shown in the following steps:

1. Configure XPS for each TX queue using the following command:

```
echo [cpumask] > /sys/class/net/$ifname/queues/tx-$i/xps_cpus
```

Note: Any application generating traffic on CPUs in the CPU mask interrupts the ring tx-$i specified here.

To configure aRFS, use the following steps:

1. Enable ntuple filtering (required for aRFS) using the following command:
   ```
   ethtool -K [interface] ntuple on
   ```
2. Configure RFS for each RX queue using the following commands:

```
echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
rps_flow_value = 32768/(number of rings)
echo [rps_flow_value] > /sys/class/net/[interface]/queues/rx-$i/rps_flow_cnt
```

Note: See the Linux kernel Scaling.txt for more details on both of these mechanisms.

TX and RX Queue Size

Provides information on TX and RX queue size and how this impacts packet drops.

Increasing the TX and RX queue size helps with queuing more data for transmit and receive and helps in avoiding packet drop during high data transfer.

Increase the TX and RX queue size to 2047 using the following command:

```
ethtool -G [interface] tx 2047 rx 2047
```

However, this is not suggested for all cases as it also results in higher latency by a bigger buildup in the software rings. There could be other side effects such as poor cache utilization if the ring size exceeds the cache size.

Interrupt Moderation

Provides information on interrupt moderation and how it impacts the rate of interrupts to the CPU.

Interrupt moderation controls the rate of interrupts to the CPU during the TX and RX. Too many interrupts (per packet interrupt) increase CPU usage, impacting the throughput adversely, while too few interrupts (after time or number of packets) increase the latency.

There are three interrupt moderation settings:

- Low – Less interrupt moderation.
- Medium – Default setting.
- High – More interrupt moderation.

Enabling adaptive-rx improves RX latency at low packet-receiving rates and improves throughput at high packet-receiving rates and therefore provides a good performance balance.

```
ethtool -C [interface] adaptive-rx on
```

GRO (Generic Receive Offload)

Provides information on GRO (Generic Receive Offload) and how it can be used to combine receive packets into a single packet.

GRO is an aggregation technique to coalesce several receive packets from a stream into a single large packet, thus saving CPU cycles as fewer packets need to be processed by the kernel. By default, GRO is accomplished in the Linux kernel, however, Broadcom NICs support Hardware GRO.

```
ethtool -K [interface] rx-gro-hw on lro off gro on
```

Broadcom NICs support the aggregation in HW and it can coexist with SW GRO.

Relaxed Ordering

Provides information on enabling relaxed ordering for improved performance.

Relaxed ordering allows packets to be retired out of order when possible. This maintains data consistency and improves performance in high bandwidth cases. Relaxed ordering can be queried and enabled using the NICCLI tool:

NICCLI Command

To query the RO status, use the following command:

```
niccli -i <index> nvm --getoption pcie_relaxed_ordering
```

To enable RO, use the following command:

```
niccli -i <index> nvm --setoption pcie_relaxed_ordering --value 0x1
```

Note: Ensure the following criteria are met before enabling RO:

- Memory accesses to the same location are in order.
- All devices between the endpoint (NIC) and the root complex support relaxed ordering.
- No applications are in use that are designed to poll on the last byte-address of the sink buffer to mark the completion for example latency-sensitive RDMA applications.

It is recommended to enable this feature if all three conditions above are satisfied. This option significantly improves the throughput on AMD EPYC 7002 for high bandwidth scenarios.

PCIe MRRS (Maximum Read Request Size)

Provides information using the PCIe MRRS (maximum read request size) to enforce uniform bandwidth allocation.

This parameter specifies the maximum size of a memory read request. The MRRS can be used to enforce a more uniform allocation of bandwidth by imposing a ceiling on the read requests. The MRRS can be queried and set dynamically using the following commands:

To identify the PCIe bus for Broadcom NICs, use the following commands:

```
lspci | grep Broadcom
41:00.0 Ethernet controller: Broadcom Limited Device 1750
```

To query the current MRRS value, use the following commands:

```
lspci -s 0000:41:00.0 -vvv | grep MaxReadReq
MaxPayload 512 bytes, MaxReadReq 4096 bytes
```

To identify the MRRS size selector, use the following commands:

```
setpci -s 41:00.0 b4.w
```

The return value is: `5d57`

The first digit (shown in the previous command example) is the MRRS size selector, and the number 5 represents the MRRS value of 4096B. Other acceptable values are as follows:

```
0 -> 128B, 1 -> 256B, 2 -> 512B, 3 -> 1024B, 4 -> 2048B and 5 -> 4096B
```

To change MRRS from 4096B, use the following commands:

```
setpci -s 41:00.0 b4.w=3d57
lspci -s 0000:41:00.0 -vvv | grep MaxReadReq
MaxPayload 512 bytes, MaxReadReq 1024 bytes
```

Note: Do not change the last three digits from the setup (d57 in the previous example), it may crash the system.

Operating System Tuning (Linux)

This section contains operating system tuning information for Linux.

IOMMU

Use IOMMUIt for IOMMU in passthrough (pt) mode. This option disables the DMA remapping (DMAR) to the memory and improves the host performance. Add `iommu=pt` in the grub file to enable this option, as shown in the following commands:

```
vi /etc/default/grub
GRUB_CMDLINE_LINUX="nofb splash=quiet console=tty0 ... iommu=pt"
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Reboot the system and ensure that `iommu=pt` is in `/proc/cmdline` by using the following command:

```
cat /proc/cmdline | grep -i iommu=pt
```

Performance Governor

The CPU frequency performance governor sets the CPU statically to the highest frequency within the borders of `scaling_min_freq` and `scaling_max_freq` for highest performance.

```
echo performance | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

Check that the CPUs are running at the highest frequency by using the following command:

```
cat /proc/cpuinfo | grep -i mhz
```

TCP Memory Configuration

Increase the memory buffer for TCP sockets. This increase can improve performance for long RTT connections by allowing more data in flight at a time when smaller buffers might not cover the bandwidth-delay product (BDP), resulting in transmission gaps. The following three values in the echo statement represent the minimum, default, and maximum buffer value for each TCP socket.

```
echo "4096 131072 268435456" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 131072 63108864" > /proc/sys/net/ipv4/tcp_wmem
```

The TCP `rmem_max` and `wmem_max` are maximum receive and send buffer sizes for the socket memory. These buffers hold and send the data until it is read by the application.

```
echo 268435456 > /proc/sys/net/core/rmem_max
echo 63108864 > /proc/sys/net/core/wmem_max
```

nohz=off

The `nohz=off` parameter is a boot time kernel parameter that disables the dyntick idle mode. In this mode, the kernel sends the timer tick periodically to all CPUs irrespective of the state and prevents the CPU from remaining idle for a long time which results in increased power consumption.

Note: This configuration must be tested extensively as results can vary depending on the workload and applications.

The following example disables the dyntick idle mode.

```
vi /etc/default/grub
GRUB_CMDLINE_LINUX="nofb splash=quiet console=tty0 ... nohz=off"
grub2-mkconfig -o /boot/grub2/grub.cfg
```

TCP Example with the BCM957508-P2100G

This section provides the BCM957508-P2100G configuration for running bi-directional dual port test with iperf3 for 2 queues and 2 sessions test.

NOTE: The commands below are for reference and may not be a complete set of commands due to brevity.

BIOS Settings

```
Configure NPS=1 Enable X2APIC
Performance Determinism Slider Configure APBDIS = 1
Configure Preferred IO and Enhanced Preferred IO Enable PCIe Ten Bit Tag
Configure Memory Clock Speed Enable* L3 as NUMA
Configure xGMI
```

* This option should be set to auto or disable for EPYC 7003 and above series processors.

Adapter Settings (NICCLI)

Note: The port 0 interface is enp65s0f0 and the port 1 interface enp65s0f1.

```
ethtool -L enp65s0f0 combined 2
ethtool -C enp65s0f0 adaptive-rx on
ethtool -G enp65s0f0 rx 2047 tx 2047
ethtool -K enp65s0f0 ntuple on
ethtool -K enp65s0f0 tx-nocache-copy on
ethtool -K enp65s0f0 rx-gro-hw on lro off gro on
niccli -i <index> nvm --setoption performance_profile --value 0
niccli -i <index> nvm --setoption pcie_relaxed_ordering --value 1
ethtool -L enp65s0f1 combined 2
ethtool -C enp65s0f1 adaptive-rx on
ethtool -G enp65s0f1 rx 2047 tx 2047
ethtool -K enp65s0f1 ntuple on
ethtool -K enp65s0f1 tx-nocache-copy on
ethtool -K enp65s0f1 rx-gro-hw on lro off gro on
niccli -i <index> nvm --setoption performance_profile --value 0
niccli -i <index> nvm --setoption pcie_relaxed_ordering --value 1
```

Operating System Settings

```
echo 268435456 > /proc/sys/net/core/rmem_max
echo 67108864 > /proc/sys/net/core/wmem_max
echo "4096 131072 268435456" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 131072 63108864" > /proc/sys/net/ipv4/tcp_wmem
```

Number of Channels (Combined Rings) Configuration

```
ethtool -L ens7f0np0 combined 2
ethtool -L ens7f1np1 combined 2
```

Configure Affinities

```
IRQ list for Port0: 229,230
IRQ list for Port1: 231,232
```

CPU Cores to be Assigned – Port 0: 1, 3 Port 1: 5, 7

```
echo 1 > /proc/irq/229/smp_affinity_list
echo 3 > /proc/irq/230/smp_affinity_list
echo 5 > /proc/irq/231/smp_affinity_list
echo 7 > /proc/irq/232/smp_affinity_list
```

ARFS and XPS Configure

Port 0:

```
ethtool -K ens7f0np0 ntuple on
echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
echo 16384 > /sys/class/net/ens7f0np0/queues/rx-0/rps_flow_cnt
echo 16384 > /sys/class/net/ens7f0np0/queues/rx-1/rps_flow_cnt
```

Port 1:

```
ethtool -K ens7f1np1 ntuple on
echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
echo 16384 > /sys/class/net/ens7f1np1/queues/rx-0/rps_flow_cnt
echo 16384 > /sys/class/net/ens7f1np1/queues/rx-1/rps_flow_cnt
```

XPS Setting on Port0

```
echo 2 > /sys/class/net/ens7f0np0/queues/tx-0/xps_cpus
echo 8 > /sys/class/net/ens7f0np0/queues/tx-1/xps_cpus
```

XPS Setting on Port1

```
echo 20 > /sys/class/net/ens7f1np1/queues/tx-0/xps_cpus
echo 80 > /sys/class/net/ens7f1np1/queues/tx-1/xps_cpus
```

Iperf Server Commands

```
taskset -c 1 iperf3 -s --port 4500 -i 1
taskset -c 3 iperf3 -s --port 4501 -i 1
taskset -c 5 iperf3 -s --port 4502 -i 1
taskset -c 7 iperf3 -s --port 4503 -i 1
```

Iperf Client Commands

```
taskset -c 1 iperf3 -c 192.168.1.1 -Z -l 64K -w 512K -t 60 -i 1 -p 4500
taskset -c 3 iperf3 -c 192.168.1.1 -Z -l 64K -w 512K -t 60 -i 1 -p 4501
taskset -c 5 iperf3 -c 172.16.1.1 -Z -l 64K -w 512K -t 60 -i 1 -p 4502
taskset -c 7 iperf3 -c 172.16.1.1 -Z -l 64K -w 512K -t 60 -i 1 -p 4503
```

Table 42: Server Configuration

Item	Description
Test	Linux Bi-Directional TCP, 2 x 100Gb/s
Server	Intel Ice Lake Dell platforms
CPU	Intel Xeon Platinum 8358 CPU at 2.60 GHz
RAM	131 GB,16 GB x 8 DIMMs at 3200 MHz
NIC	BCM957508-P2100G
Operating System	Red Hat Enterprise Linux release 8.3
Test Tool	Iperf3
Test Type	Linux Single-ended iperf (Broadcom Automation) on servers connected back-to-back.

Configuring RSS for Performance in Windows

Provides the steps required to configure NetPerf sessions to fully use the receive queues in Windows.

When running single-ended traffic, ensure that all cores are used equally for the best performance. One way to achieve this is to ensure all traffic is spread across all available queues using RSS.

For an introduction to RSS, refer to the following Microsoft site: https://msdn.microsoft.com/en-us/windows/hardware/drivers/network/ introduction-to-receive-side-scaling

This section describes how to monitor the receive queues using Windows PowerShell and how to ensure that all queues are used. PowerShell automates this methodology. If a graphical interface is preferred, the Performance Monitor displays the same results. This section uses NetPerf for examples.

This section provides the following information on RSS for Performance in Windows:

- Setting Up RSS
- Monitoring Receive Queues/Cores
- Expanding to New Counters
- Configuring RSSv2

Setting Up RSS

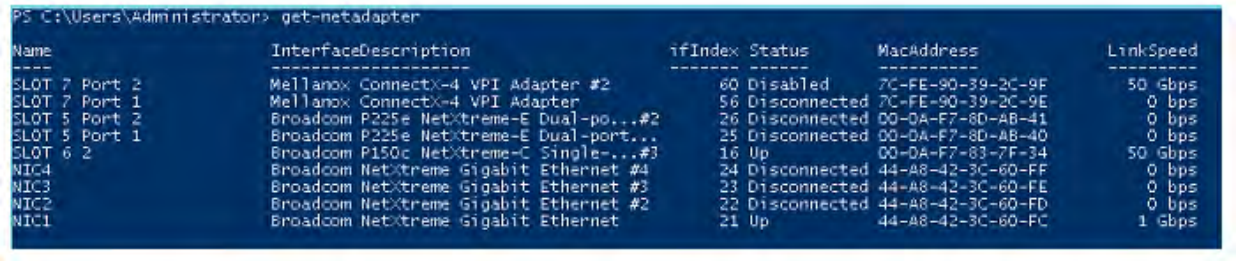
Provides information on setting up RSS on Ethernet network adapters.

Set up the controller so that the proper cores are used using the following steps:

Note: This process depends on the target test environment. The purpose of this section is to provide steps for manipulating the RSS settings. The value settings used in the examples are not necessarily the ideal settings for achieving optimal performance.

1. To show the available adapters and their names used in subsequent steps, use the following command:

   ```
   Get-Netadapter
   ```



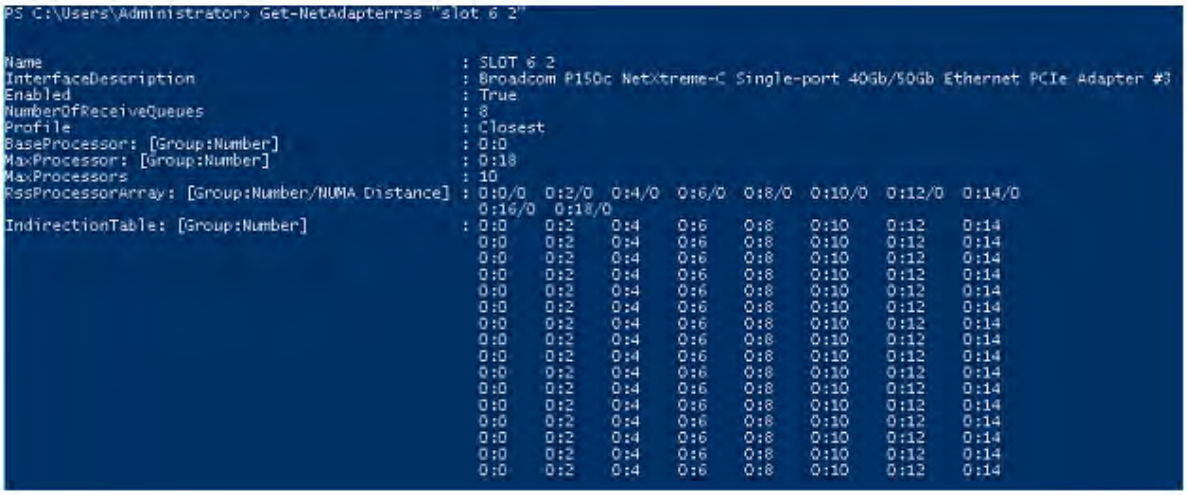
```
PS C:\Users\Administrator> get-netadapter

Name                      InterfaceDescription                ifIndex Status       MacAddress             LinkSpeed
----                      --------------------                ------- ------       ----------             ---------
SLOT 7 Port 2             Mellanox ConnectX-4 VPI Adapter #2       60 Disabled     7C-FE-90-39-2C-9F       50 Gbps
SLOT 7 Port 1             Mellanox ConnectX-4 VPI Adapter          56 Disconnected 7C-FE-90-39-2C-9E        0 bps
SLOT 5 Port 2             Broadcom P225e NetXtreme-E Dual-po...#2  26 Disconnected 00-0A-F7-8D-AB-41        0 bps
SLOT 5 Port 1             Broadcom P225e NetXtreme-E Dual-port...  25 Disconnected 00-0A-F7-8D-AB-40        0 bps
SLOT 6 2                  Broadcom P150c NetXtreme-C Single-...#3  16 Up           00-0A-F7-83-7F-34       50 Gbps
NIC4                      Broadcom NetXtreme Gigabit Ethernet #4   24 Disconnected 44-A8-42-3C-60-FF        0 bps
NIC3                      Broadcom NetXtreme Gigabit Ethernet #3   23 Disconnected 44-A8-42-3C-60-FE        0 bps
NIC2                      Broadcom NetXtreme Gigabit Ethernet #2   22 Disconnected 44-A8-42-3C-60-FD        0 bps
NIC1                      Broadcom NetXtreme Gigabit Ethernet      21 Up           44-A8-42-3C-60-FC        1 Gbps
```

   For this example, Slot 6 2 is used, which indicates a Broadcom single-port 50 Gb/s adapter.

2. To return all available RSS properties, use the following command:

```
Get-NetAdapterRss slot 6 2.
```



The indirection table is what must be manipulated. The indirection table indicates where the interrupt is processed for each hash value. If the hash returns 0, processor 0 is used. If the hash returns 1, processor 2 is used, and so forth.

Note: Windows displays even-numbered processors because hyperthreading is enabled. Windows does not use both cores in a hyperthreaded pair. Therefore, only physical cores are displayed. On systems with hyperthreading off, the physical cores are sequentially numbered (for example, 0, 1, 2, 3).

3. Ensure that the Virtual Receive-Side scaling is enabled with the following command:

```
Set-NetadapterRss "slot 6 2" -Enabled $False
```

4. Ensure that the Profile is set to **Closest** with the following command:

```
Set-NetadapterRss "slot 6 2" -Profile "Closest"
```

Profiles

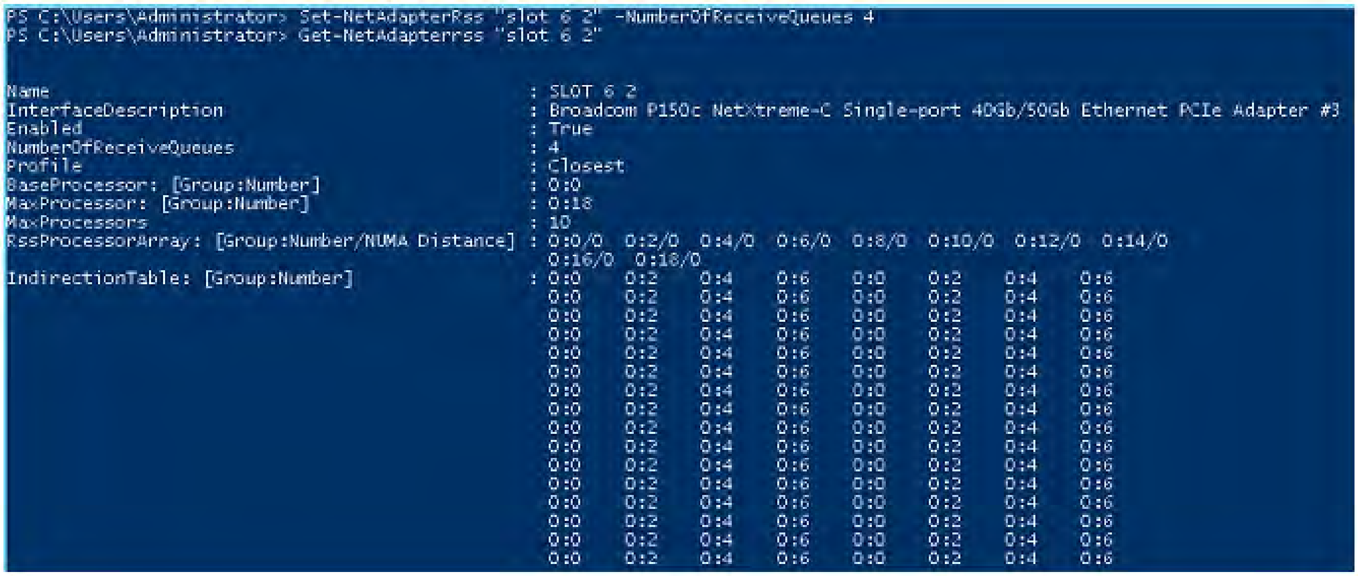
RSS profiles determine how CPUs are assigned to the NIC. Select from five profiles; the profile affects how the indirection table is populated. The following profiles are currently available RSS profiles. For some profiles, the operating system can dynamically load balance. If the operating system is load balancing, ensure that the RssProcessorArray only shows the core intended for RSS use. To change this list, use BaseProcessor, MaxProcessor, and MaxProcessors to limit the list.

- Closest – Logical processor numbers that are near the network adapter's base RSS processor are preferred. With this profile, the operating system rebalances logical processors dynamically based on load.
- ClosestStatic – Logical processor numbers near the network adapter's base RSS processor are preferred. With this profile, the operating system does not rebalance logical processors dynamically based on load.
- NUMA – Logical processor numbers are generally selected on different NUMA nodes to distribute the load. With this profile, the operating system rebalances logical processors dynamically based on load.

- NUMAStatic – Default profile. Logical processor numbers are generally selected on different NUMA nodes to distribute the load. With this profile, the operating system does not rebalance logical processors dynamically based on load.
- Conservative – RSS uses as few processors as possible to sustain the load. This option helps reduce the number of interrupts.

1. To set the number of receive queues, use the following command:

```
Set-NetAdapterRss "slot 6 2" -NumberOfReceiveQueues 4
```



```
PS C:\Users\Administrator> Set-NetAdapterRss "slot 6 2" -NumberOfReceiveQueues 4
PS C:\Users\Administrator> Get-NetAdapterrss "slot 6 2"

Name                                            : SLOT 6 2
InterfaceDescription                            : Broadcom P150c NetXtreme-C Single-port 40Gb/50Gb Ethernet PCIe Adapter #3
Enabled                                         : True
NumberOfReceiveQueues                           : 4
Profile                                         : Closest
BaseProcessor: [Group:Number]                   : 0:0
MaxProcessor: [Group:Number]                    : 0:18
MaxProcessors                                   : 10
RssProcessorArray: [Group:Number/NUMA Distance] : 0:0/0  0:2/0  0:4/0  0:6/0  0:8/0  0:10/0  0:12/0  0:14/0
                                                  0:16/0  0:18/0
IndirectionTable: [Group:Number]                : 0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
                                                  0:0   0:2   0:4   0:6   0:0   0:2   0:4   0:6
```

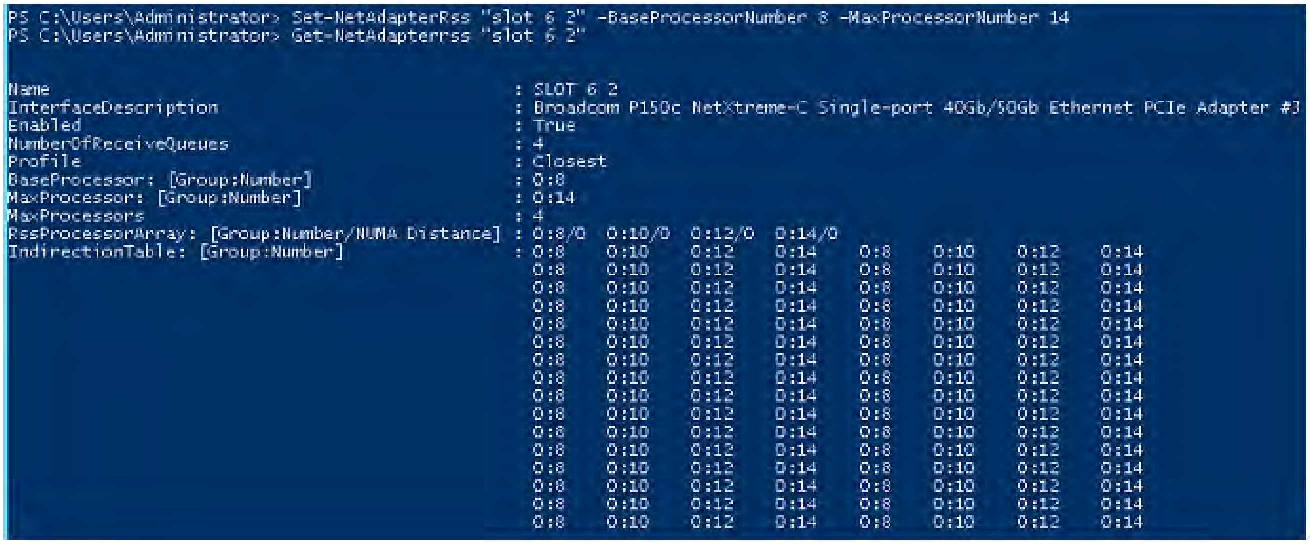
Note: The indirection table has changed. Only four queues exist instead of eight, therefore, only four cores are used.

2. To set the base processor and max processor, use the following command:

```
Set-NetAdapterRss "slot 6 2" -BaseProcessorNumber 8 -MaxProcessorNumber 14
```

This command effectively changes the adapter's RSS processor group, RssProcessorArray.

Note: The current assumption is that the processors are in group 0. To change to group 1 use BaseProcessorGroup and MaxProcessorGroup.



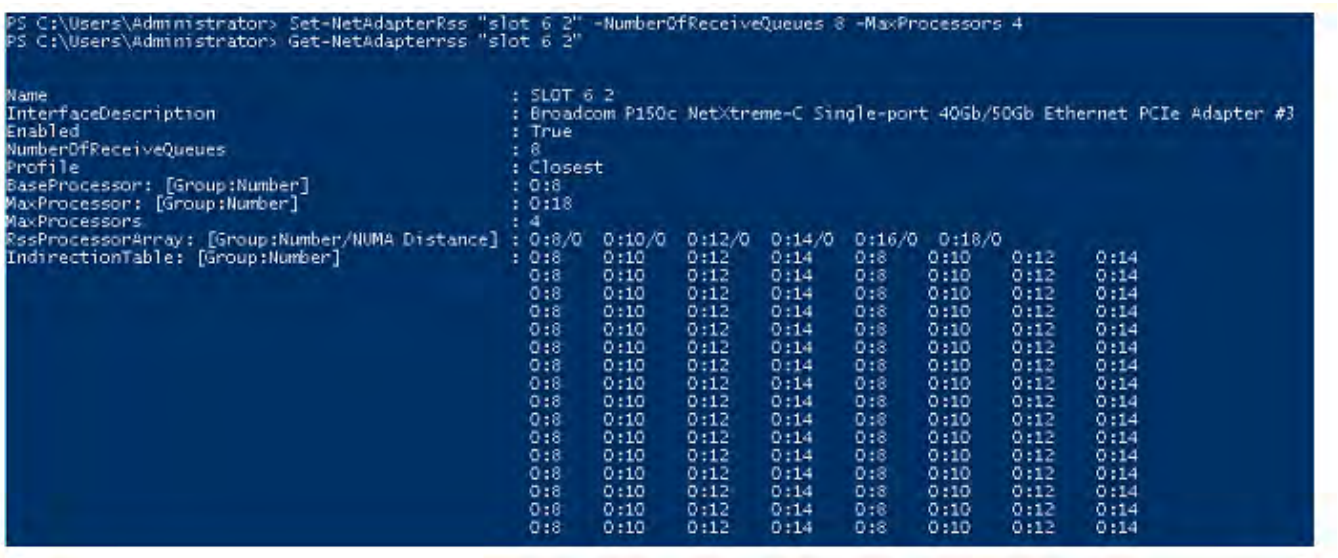
```
PS C:\Users\Administrator> Set-NetAdapterRss "slot 6 2" -BaseProcessorNumber 8 -MaxProcessorNumber 14
PS C:\Users\Administrator> Get-NetAdapterrss "slot 6 2"

Name                                            : SLOT 6 2
InterfaceDescription                            : Broadcom P150c NetXtreme-C Single-port 40Gb/50Gb Ethernet PCIe Adapter #3
Enabled                                         : True
NumberOfReceiveQueues                           : 4
Profile                                         : Closest
BaseProcessor: [Group:Number]                   : 0:8
MaxProcessor: [Group:Number]                    : 0:14
MaxProcessors                                   : 4
RssProcessorArray: [Group:Number/NUMA Distance] : 0:8/0   0:10/0   0:12/0   0:14/0
IndirectionTable: [Group:Number]                : 0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
                                                  0:8   0:10   0:12   0:14   0:8   0:10   0:12   0:14
```

Note: In the RssProcessorArray, the operating system is now limited to only using four processors (8 through 14 non-hyperthreaded cores) as the base processor is set to 8 with a max processor of 14.

3. To limit the number of cores in the operating system to four, use the following command:

```
Set-NetAdapterRss "slot 6 2" -NumberOfReceiveQueues 8 -MaxProcessors 4.
```



Note: In the indirection table, there are eight queues to service and six cores are in the ProcessorArray, however, MaxProcessors is set to four. The indirection table is only populated with four processors, limiting it to four queues.

Monitoring Receive Queues/Cores

Provides information on monitoring receive queues/cores for expected hashes.

The counters monitor where each flow is received. A single stream is started with a static MAC/IP/Port number (for example, the TCP Port number is specified and does not allow the operating system to choose the port). The Received Packets/sec value on each core is monitored. The expectation is that only one core receives the packets per stream which indicates where the stream is being hashed. As long as the driver is not reloaded, this stream is always hashed the same.

To monitor the receive queues/cores, use the following instructions.

1. `Get-Counter -counter "\Per processor network interface card activity(*, {Adapter Description})\Received Packets/sec".`
   Example: `Get-Counter -counter "\Per processor network interface card activity(*, Broadcom P150c Single-port 40Gb-50Gb Ethernet PCIe Adapter #3)\Received Packets/sec"`
   This command monitors the Received Packets/sec for the cores associated with the 50G Broadcom NIC. To get the specific count of an individual processor, change the `*` to the processor number (for example, 8 or 10 or 12 or 14).
2. The command in Step 1 executes a single time and returns the results. To execute the command multiple times, use the following flags in the command:
   - `SampleInterval`
   - `MaxSamples`
   - `Continuous`
3. If the proper adapter string is not known, find it by using the IP address:

```
$ipaddr = "192.168.1.21"
```

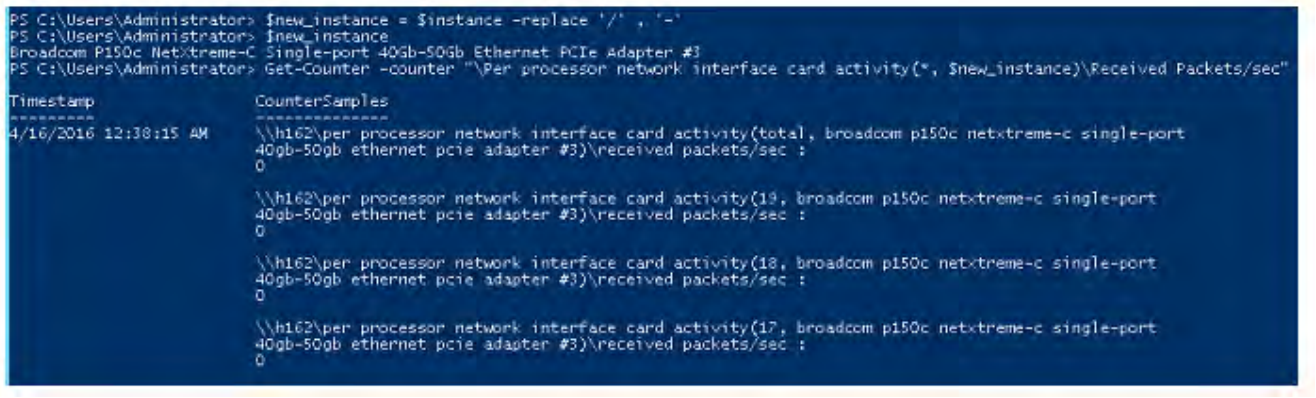
```
$instance = Get-WmiObject -Class Win32_NetworkAdapterConfiguration |Where-Object {$_.IPAddress - contains
 $ipaddr}|Select-Object -expand Description
```

`$instance` now holds the string of the description of the adapter with the given IP address. This command assumes that exactly one adapter has the given IP address.

```
PS C:\Users\Administrator> $ipaddr
192.16.1.21
PS C:\User\Administrator> $instance + Get-WmiObject -Class Win32_NetworkAdapterConfiguration [where-Object
 {$_.IPAddress -contains $ipaddr}] Select-Object -expand Description
PS C:\Users\Administrator> $instance
Broadcom P150C NetXtreme-C Single-port 40Gb/50Gb Ethernet PCIe Adapter #3
```

Note: When switching to 50Gb from 40Gb, the command uses / and the tool expects a dash (-). The following command corrects this issue.

4.
```
$new_instance = $instance -replace '/' , '-'
Get-Counter -counter "\Per processor network interface card activity(*, $new_instance)\Received Pack-
ets/sec"
```



   a. The following command reports the same information. That is, interrupts/sec from the per processor network interface card activity object.

```
Get-Counter -counter "\Per processor network interface card activity(*, $new_instance)\Interrupts/ sec"
```

   b. The interrupts/sec from the processor object (this does not allow filtering based on a specific network adapter).

```
Get-Counter -counter "\Processor(*)\Interrupts/sec"
```

5. Using the `Get-Counter` cmdlet, cores can be monitored while transmitting traffic. As each stream is transmitted, monitor the receive packets to understand which queue is being utilized. When a stream that exercises a new unique queue is found, record it. If the stream exercises an already used queue, try a new stream with a different DATA Port.

   Note: In NetPerf, the data ports are the ports that matter in this case, not the control port.

6. To send traffic using NetPerf, use the following command:

```
netperf.exe -l 10 -H 192.16.1.21 -p 12801 -P 0 -- -S 512K-m 64k-s 512K -P 32212,32214
```

   Note: 12801 is the NetPerf control port. 32212 is the control port on the transmit side while 32214 is the data port on the remote side.

7. On the receive side, run the following command to monitor the receive packets per core. Only cores that receive packets report. This command is run 20 times:

```
1..20 | % { (Get-Counter -counter "\Per processor network interface card activity(*,
```

```
$new_instance)\Received Packets/sec").countersamples|Where-Object cookedvalue -ne 0| Format-Table Instan-
ceName, CookedValue -auto}
```

```
InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 82807.6198989239
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    82820.6569945336
12, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    4.01141403376079



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 84983.9050153597
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    84983.9050153597



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 90142.5377507455
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    90143.5382117587



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 90191.9038597847
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    90192.9057694032
```

Note: Initially, two cores receive packets. The first in red (12) is the control port setting up the test. The second core (14) is the data port. Additional packets are reported after initiation. Ensure that this transmit command resolves to core 14.

8. To resolve multiple streams, try a new stream with a different data port where it does not resolve to the same core as previously (for example, core 14).

```
netperf.exe -l 10 -H 192.16.1.21 -p 12802 -P 0 -- -S 512K-m 64k-s 512K -P 32223,32225
```

```
InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 56534.9190857588
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    56538.9596753266
12, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    4.04058956783525



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 82041.4098797662
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    82043.5196365778



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 82268.8682196618
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    82268.8682196618



InstanceName                                                                CookedValue
------------                                                                -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 81213.4623495728
14, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3     81211.102012006
```

Because this stream also resolves to core 14, it cannot be used. A new data port must be tried.

```
netperf.exe -l 10 -H 192.16.1.21 -p 12802 -P 0 -- -S 512K-m 64k-s 512K -P 32245,32247
```

```
InstanceName                                                                    CookedValue
------------                                                                    -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 83011.7197160041
10, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    83005.7000846449



InstanceName                                                                    CookedValue
------------                                                                    -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 88902.8878203428
10, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    88909.8964044222



InstanceName                                                                    CookedValue
------------                                                                    -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 90201.2954833553
10, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    90202.2973192374



InstanceName                                                                    CookedValue
------------                                                                    -----------
total, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3 90199.1241305563
10, broadcom p150c netxtreme-c single-port 40gb-50gb ethernet pcie adapter #3    90201.1273091148
```

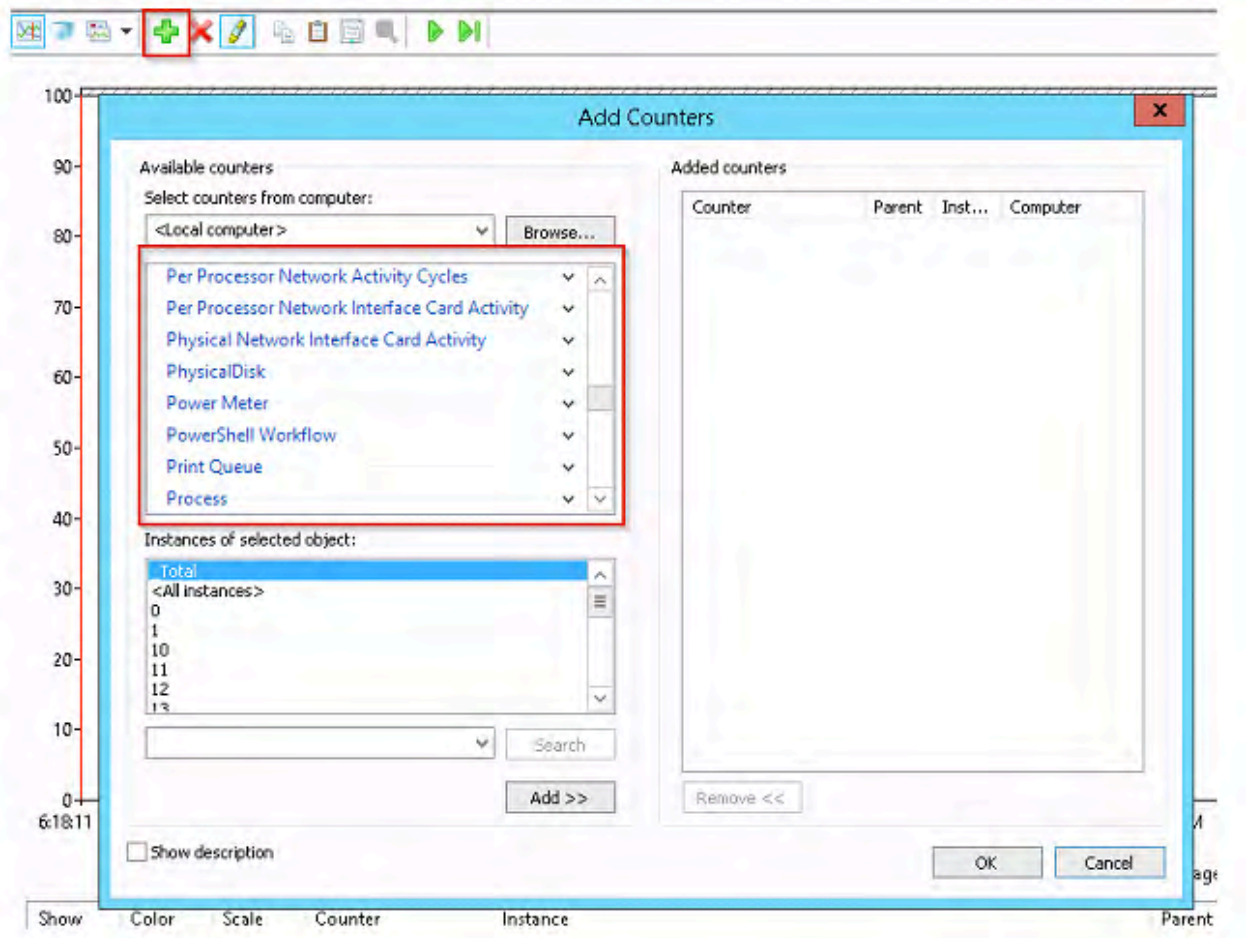
This stream uses core 10 and can be kept. Two streams are resolved to different cores. Continue this process until all cores are resolved.

Expanding to New Counters

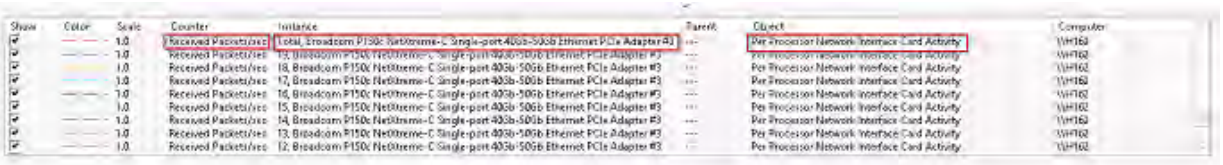
Provides information on expanding new counters for monitoring.

Additional or different counters must be monitored. There are several ways to discover what is available. Query using PowerShell or open Performance Monitor.

1. After the application is open, click the **+** button as shown in the following figure. This action displays every available counter.



2. For this example, select **Per Processor Network Interface Card Activity** and from the drop-down menu **Received Packets/sec**.
3. Select an instance to monitor from **Instances of selected object**.
4. Select **<All instances>**for this example.
5. After the **Counter and Instance are highlighted**, click **Add>>**. An item displays in **Added Counters**.
6. Click **OK**.
   A large window on top displays the monitor and a small window displays each counter instance.
7. For debugging, run the test and visually monitor the counters. Select show/hide, scale, highlight counters, and several other items.
8. Monitor the instances window. Three columns must be extracted: Counter, Instance, and Object.



   a. Object is the most important column. This is the first item selected previously in the top left window. An object contains several counters.
   b. Counter is the next in importance and displays the specific counter monitored within the object.
   c. Instance is another level of granularity that is monitored.

For this specific counter, the instance is a tuple. The first value is the processor and the second is the adapter. For example, if only core 0 was monitored for this particular NIC, monitor 0, Broadcom P150c Single-port 40Gb-50Gb Ethernet PCIe Adapter #3 is the result.

Configuring RSSv2

Provides information on configuring RSSv2.

Performance-enhancing features such as RSSv2 can be enabled with a registry key. Broadcom network adapter parameters are located in the registry under the following registry key:

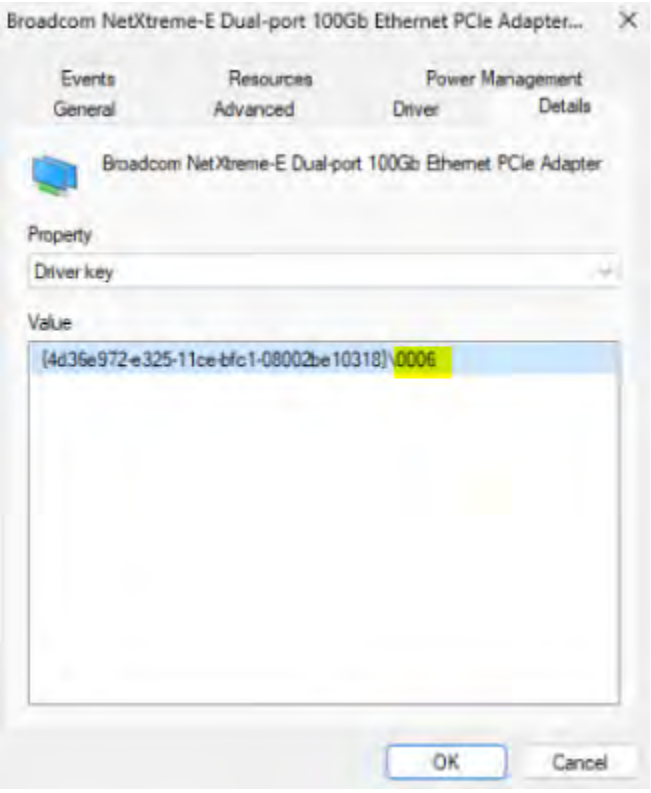
```
HKEY_LOCAL_MACHINE
\SYSTEM\CurrentControlSet
\ Control
\ Class
\{4d36e972-e325-11ce-bfc1-08002be10318}
\<Index>
```

Obtaining the Index Number

To obtain the index number:

1. Open **Device Manager>Network Adapters**.
2. Right-click on **Properties** on the Broadcom NetXtreme Ethernet Adapter
3. Click **Details.**
4. Select **Driver Key** and obtain the index number at the end of the string as shown in the following figure.

Figure 33: Driver Key



Editing the Registry

To edit the registry to add the rssv2 value:

1. On the right-hand panel, right-click and select **New**.
2. Select **DWORD**.
3. Name the value **rssv2**.
4. Set the vlue to **1** to enable.
5. Set the vlue to **0** to disable.
6. Exit the registry.
7. Restart the driver by disabling/enabling the interface.

DPDK Tuning for Ethernet Network Adapters

Provides tuning information for improving DPDK performance for Ethernet network adapters.

Broadcom publishes the NIC Data Plane Development Kit (DPDK) performance report to dpdk.org. This report contains the latest performance numbers and configuration details. The most recent version of the report can be accessed using the following link:

https://core.dpdk.org/perf-reports/

This section provides the following information on DPDK tunings:

- BIOS Tuning
- Kernel Tuning
- Adapter Tuning
- Application Tuning
- DPDK Results

TCP BIOS Performance Tuning

Provides information on setting BIOS options for improved performance.

See BIOS Tuning and set the following BIOS options:

Intel Skylake (Gen3)

- HyperThreading/Logical Cores – Disable
- System Profile – Performance

AMD Milan (Gen4)/AMD Genoa (Gen5)

- Logical Processor – Disabled
- IOMMU – Enabled
- NPS – 1
- L3 Cache as NUMA – Disabled
- X2APIC Mode – Enabled
- PCIe Ten Bit Tag – Enabled
- Preferred IO Bus – Enabled (Milan Only)
- Preferred IO Bus Value – Provide BUS ID (Milan Only)
- Enhanced Preferred IO – Enabled (Milan Only)
- CPU Power Management – Maximum Performance
- Memory Frequency – Maximum Performance

- Determinism Slider – Performance
- APBDIS – Enabled
- xGMI Link Width Control – Manual
- xGMI Force Link Width – 2
- xGMI Force Link Width Control – Force
- xGMI Max Link Width Control – Auto

Intel Icelake (Gen4)/Intel Sapphire Rapids (Gen5)

- Logical Processor – Disabled
- Virtualization Technology – Enabled
- Sub NUMA Cluster – Disabled
- MADT Core Enumeration – Round Robin
- I/O Snoop HoldOff Response – 2K Cycles
- SR-IOV Global Enable – Disabled
- System Profile – Performance
- x2APIC Mode – Enabled

Kernel Tuning

Provides instructions on modifying the kernel for improved performance.

Intel Skylake (Gen3) Boot Settings

```
isolcpus=3-33 nohz_full=3-33 rcu_nocbs=3-33 default_hugepagesz=1G hugepagesz=1G hugepages=64 intel_idle.max_c-
state=0 processor.max_cstate=0 intel_pstate=disable rcu_nocb_poll audit=0 nosoftlockup intel_iommu=on iommu=pt
 mce=ignore_ce
```

AMD Milan (Gen4) Boot Settings

```
default_hugepagesz=1G hugepagesz=1G hugepages=64 nosoftlockup processor.max_cstate=0 amd_iommu=on iommu=pt nu-
ma_balancing=disable selinux=0 nohz=off rcu_nocb_poll audit=0 mce=ignore_ce isolcpus=1-36 rcu_nocbs=1-36
```

Intel Icelake (Gen4) Boot Settings

```
default_hugepagesz=1G hugepagesz=1G hugepages=64 intel_idle.max_cstate=0
 processor.max_cstate=0 intel_pstate=disable rcu_nocb_poll audit=0 nosoftlock-
up mce=ignore_ce isolcpus=4,7,8,11,12,15,16,19,20,23,24,27,28,31,32,35,36,39
 nohz_full=4,7,8,11,12,15,16,19,20,23,24,27,28,31,32,35,36,39 rcu_nocb-
s=4,7,8,11,12,15,16,19,20,23,24,27,28,31,32,35,36,39 intel_iommu=on iommu=pt
```

Adapter Tuning

Provides information on setting adapter options for improved performance

Enable Relaxed Ordering (for AMD/Gen4 only)

NICCLI Commands

To Enable:

```
niccli -i <index> nvm --setoption pcie_relaxed_ordering --value 1
```

To Verify:

```
niccli -i <index> nvm --getoption pcie_relaxed_ordering
```

Disable LLDP

To Disable LLDP, use the following commands:

```
niccli -i <index> nvm --setoption lldp_nearest_bridge --scope 0 --value 0
niccli -i <index> nvm --setoption lldp_nearest_bridge --scope 1 --value 0
niccli -i <index> nvm --setoption lldp_nearest_non_tpmr_bridge --scope 0 --value 0
niccli -i <index> nvm --setoption lldp_nearest_non_tpmr_bridge --scope 1 --value 0
```

Disable RDMA

To Disable RDMA, use the following commands:

```
niccli -i <index> nvm --setoption support_rdma --scope 0 --value 0
niccli -i <index> nvm --setoption support_rdma --scope 1 --value 0
```

Performance Profile

To set the default profile:

```
niccli -i <index> nvm --setoption performance_profile --value 0
```

Application Tuning

Provides information on setting application options for improved performance

Driver: vfio-pci

Execute `testpmd` with realtime scheduling priority:

```
# echo -1 > /proc/sys/kernel/sched_rt_runtime_us
# chrt -r 1 ./testpmd -l 4,6,8 -n4  -- --txq=2 --rxq=2 --nb-cores=2 -i --burst=64 -a
```

testpmd command line:

```
chrt -r 1 ./dpdk-testpmd -l 4,5,6,7,8 --main-lcore 4 -n 4 -a 41:00.0 -a 41:00.1 -- -i --nb-cores=4 --nb-
ports=2 --rxq=4 --txq=4 --rxd=2048 --txd=2048 --socket-num=0 --burst=64 --record-core-cycles --record-burst-
stats -a
```

DPDK Results

Provides information on interpreting results for DPDK.

- http://core.dpdk.org/perf-reports/
- BCM5741X (25G) Results:
  - Forwarding Rate is 34 Mp/s using 64B frame
  - Line-Rate from 128B onwards
- BCM575XX (200G) Results:
  - Forwarding Rate is 106 Mp/s using 64B frame
  - Line-Rate with 512B

IP Forwarding Tunings for Ethernet Network Adapters

Provides tuning information to improve IP forwarding performance for Ethernet network adapters.

This section provides the following information on IP forwarding tunings:

- BIOS Tuning
- Kernel Tuning
- Adapter Tuning
- IP Forwarding Results

BIOS Tuning

Provides tuning information for IP forwarding for Ethernet network controllers.

This section provides information on BIOS tuning.

- SVM Mode – Disabled
- SMEE – Disabled
- SR-IOV Support – Disabled
- Custom Pstate0 – Auto
- Custom Pstate1 – Disabled
- Custom Pstate2 – Disabled
- SMT Control – Disabled
- Local APIC Mode – x2APIC
- NUMA nodes per socket – NPS4
- DDR Timing Configuration → Overclock – Enabled
- Memory Clock Speed – 2666 MHz
- DDR Power Options → Power Down Enable – Disabled
- IOMMU – Auto
- Determinism Control – Manual
- Determinism Slider – Performance

Kernel Tuning

Provides instructions for tuning the kernel for improved performance.

- Add the following entries to the kernel command line:
  amd_iommu=off iommu=off nohz=off (for 100G Link, set iommu=pt)
- Map Interrupts to CPUs (CPUs that belong to local NUMA, one IRQ per CPU)
- Disable the following services
  - Firewalld
    ```
    systemctl stop firewalld
    ```
  - Selinux
    ```
    echo 0 > /selinux/enforce
    ```
  - Set CPU to run at max frequency
    ```
    echo performance | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
    ```
  - irqbalance
    ```
    systemctl stop irqbalance
    ```
  - NetworkManager

    ```
    systemctl stop NetworkManager
    ```

- Ensure NAT/IP Table modules are unloaded (list of modules that need to be unloaded)
  - iptable_raw
  - iptable_security
  - kvm_amd
  - ip6table_mangle
  - ip6table_security
  - ip6table_raw
  - iptable_mangle
  - iptable_filter
  - ip_tables
  - ip6table_filter
  - ip6_tables
  - ipt_REJECT
  - ebtable_nat
  - ebtable_filter
  - ebtables
  - kvm_intel
  - Kvm

Equation 1:

Open file `/etc/modprobe.d/blacklist.conf` and turn off auto load using below syntax alias driver-name off

Note: Deny list the dependent modules that do not unload any of the previous modules.

Adapter Tuning

Provides information on tuning the adapter for improved performance.

- Increase combined rings to 16 (1 ring per physical core in the local CCD, determine local cores using lscpu commands):
  ```
  ethtool -L [interface] combined 16 rx 0 tx 0
  ```
- Disable Pause:
  ```
  ethtool -A [interface] tx off rx off
  ```
- Disable LRO and GRO:
  ```
  ethtool -K [interface] rx-gro-hw off lro off gro off
  ```
- Turn ON TX no cache copy:
  ```
  ethtool -K [interface] tx-nocache-copy on
  ```
- Increase TX/RX Ring size to 2047:
  ```
  ethtool -G [interface] tx 2047 rx 2047
  ```
- Configure ntuple filter (to have even distribution across all rings):
  ```
  ethtool -N [interface] flow-type [udp4/tcp4] src-ip [sip] dst-ip [dip] src-port [sport] dst-port [dport]
   action [Queue to redirect]
  ```
- Interrupt Moderation (Not required for 25G):

```
ethtool -C [interface] rx-usecs 512 tx-usecs 512 rx-frames 512 tx-frames 512
```

IP Forwarding Results

Provides information on interpreting results for IP forwarding.

- IP Forwarding is typically limited by the Linux kernel. Therefore, the results scale with the number of physical cores utilized. It is common to expect roughly 600K – 800 KP/s per physical core utilized.
- BCM5741X (25G) Results:
  - Forwarding Rate is ~18 MP/s using 64B frame
  - Line-Rate from 256B onwards
- BCM575XX (100G) Results:
  - Forwarding Rate is ~18 MP/s using 64B frame

Gathering Statistics for Ethernet Network Adapters

Provides information on gathering statistics for Ethernet network adapters to verify system performance.

This section describes the following collection of statistics related to the operation of IP and RoCE:

- Ethernet Statistics
- RoCE Statistics
- RoCE Counter Definitions
- Performance Counters for Windows
- Ethtool Counters

Ethernet Statistics

Provides instructions for using ethtool to retrieve statistics for Ethernet, IP, and RoCE.

There are two types of Ethernet statistics available:

- Linux Statistics
- Windows Statistics

Linux Statistics

Provides Ethernet statistics in Linux.

Note: See Counters for additional information.

The Linux ethtool utility allows inspection of statistics for all traffic types (Ethernet, IP, RoCE) with its `-S` option. Ethtool counters include RoCE traffic, but the traffic is not specifically separated. However, RoCE and non-RoCE traffic can be differentiated by CoS queue and priority. Non-RoCE traffic is by default on priority 0 and COS queue 4. RoCE traffic is by default on priority 3 and CoS queue 0. The following example is shown for the p1p1 interface:

```
$ ethtool -S p1p1

rx_bytes_cos0: 3172896188
rx_packets_cos0: 51175745
…
rx_bytes_cos4: 1542996
rx_packets_cos4: 21330
…
rx_total_discard_pkts: 0
tx_total_discard_pkts: 0
…
rx_pfc_frames: 0
tx_pfc_frames: 0
…
rx_pause_frames: 0
tx_pause_frames: 0
```

Windows Statistics

Provides Ethernet statistics in Windows.

Windows maintains standard statistics for Ethernet Adapters. They can be checked by using `Get-NetAdapterStatistics` as shown below:

```
PS C:\Users\Administrator> Get-NetAdapterStatistics "SLOT 1 5 Port 2" | fl *


ifAlias                  : SLOT 1 5 Port 2
InterfaceAlias           : SLOT 1 5 Port 2
ifDesc                   : Broadcom P225p NetXtreme-E Dual-port 10Gb/25Gb Ethernet PCIe Adapter #4
Caption                  : MSFT_NetAdapterStatisticsSettingData 'Broadcom P225p NetXtreme-E Dual-port
 10Gb/25Gb
                           Ethernet PCIe Adapter #4'
Description              : Broadcom P225p NetXtreme-E Dual-port 10Gb/25Gb Ethernet PCIe Adapter #4
ElementName              : Broadcom P225p NetXtreme-E Dual-port 10Gb/25Gb Ethernet PCIe Adapter #4
InstanceID               : {233F1706-8CF5-4970-836F-ED985C1E4AEA}
InterfaceDescription     : Broadcom P225p NetXtreme-E Dual-port 10Gb/25Gb Ethernet PCIe Adapter #4
Name                     : SLOT 1 5 Port 2
Source                   : 2
SystemName               : XXXXX
OutboundDiscardedPackets : 0
OutboundPacketErrors     : 0
RdmaStatistics           : MSFT_NetAdapter_RdmaStatistics
ReceivedBroadcastBytes   : 0
ReceivedBroadcastPackets : 0
ReceivedBytes            : 0
ReceivedDiscardedPackets : 0
ReceivedMulticastBytes   : 0
ReceivedMulticastPackets : 0
ReceivedPacketErrors     : 0
ReceivedUnicastBytes     : 0
ReceivedUnicastPackets   : 0
RscStatistics            : MSFT_NetAdapter_RscStatistics
SentBroadcastBytes       : 14812
SentBroadcastPackets     : 56
SentBytes                : 23979
SentMulticastBytes       : 9167
SentMulticastPackets     : 87
SentUnicastBytes         : 0
SentUnicastPackets       : 0
SupportedStatistics      : 4163583
PSComputerName           :
CimClass                 : ROOT/StandardCimv2:MSFT_NetAdapterStatisticsSettingData
CimInstanceProperties    : {Caption, Description, ElementName, InstanceID...}
CimSystemProperties      : Microsoft.Management.Infrastructure.CimSystemProperties
```

The Broadcom driver provides additional proprietary statistics as shown below:

```
PS C:\Users\Administrator> Get-WmiObject -Namespace root\wmi BRCM_PortStats
```

```
__GENUS                          : 2
__CLASS                          : BRCM_PortStats
__SUPERCLASS                     :
__DYNASTY                        : BRCM_PortStats
__RELPATH                        : BRCM_PortStats.InstanceName="Broadcom P225p NetXtreme-E Dual-port
 10Gb/25Gb
                                   Ethernet PCIe Adapter #3"
__PROPERTY_COUNT                 : 131
__DERIVATION                     : {}
__SERVER                         : XXXXX
__NAMESPACE                      : root\wmi
__PATH                           : \\XXXXX\root\wmi:BRCM_PortStats.InstanceName="Broadcom P225p NetXtreme-E
                                   Dual-port 10Gb/25Gb Ethernet PCIe Adapter #3"
Active                           : True
InstanceName                     : Broadcom P225p NetXtreme-E Dual-port 10Gb/25Gb Ethernet PCIe Adapter #3
rx_1024B_1518_frames             : 0
rx_128B_255B_frames              : 88
rx_1519B_2047B_frames            : 0
rx_2048B_4095B_frames            : 0
rx_256B_511B_frames              : 0
rx_4096B_9216B_frames            : 0
rx_512B_1023B_frames             : 0
rx_64B_frames                    : 0
rx_65B_127B_frames               : 162
rx_9217B_16383B_frames           : 0
rx_align_err_frames              : 0
rx_bcast_frames                  : 0
rx_bytes                         : 33823
rx_cnp_bytes                     : 108847364832208
rx_cnp_pkts                      : 0
rx_code_err_frames               : 0
rx_ctrl_frames                   : 0
rx_double_tagged_frames          : 0
rx_eee_lpi_duration              : 0
rx_eee_lpi_events                : 0
rx_false_carrier_frames          : 0
rx_fcs_err_frames                : 0
rx_frag_frames                   : 0
rx_good_frames                   : 250
rx_good_vlan_frames              : 0
rx_hcfc_msgs                     : 0
rx_hcfc_msgs_with_crc_err        : 0
rx_jbr_frames                    : 0
rx_llfc_logical_msgs             : 0
rx_llfc_msgs_with_crc_err        : 0
rx_llfc_physical_msgs            : 0
rx_match_crc_frames              : 0
rx_mcast_frames                  : 250
rx_mtu_err_frames                : 0
rx_oor_len_frames                : 0
rx_ovrsz_frames                  : 0
rx_pause_frames                  : 0
```

```
rx_pfc_ena_frames_pri0            : 0
rx_pfc_ena_frames_pri1            : 0
rx_pfc_ena_frames_pri2            : 0
rx_pfc_ena_frames_pri3            : 0
rx_pfc_ena_frames_pri4            : 0
rx_pfc_ena_frames_pri5            : 0
rx_pfc_ena_frames_pri6            : 0
rx_pfc_ena_frames_pri7            : 0
rx_pfc_frames                     : 0
rx_pfc_xon2xoff_frames_pri0       : 0
rx_pfc_xon2xoff_frames_pri1       : 0
rx_pfc_xon2xoff_frames_pri2       : 0
rx_pfc_xon2xoff_frames_pri3       : 0
rx_pfc_xon2xoff_frames_pri4       : 0
rx_pfc_xon2xoff_frames_pri5       : 0
rx_pfc_xon2xoff_frames_pri6       : 0
rx_pfc_xon2xoff_frames_pri7       : 0
rx_promiscuous_frames             : 0
rx_rocev1_bytes                   : 108847364832208
rx_rocev1_pkts                    : 0
rx_rocev2_bytes                   : 108847364832208
rx_rocev2_pkts                    : 73014448134
rx_runt_bytes                     : 0
rx_runt_frames                    : 0
rx_sch_crc_err_frames             : 0
rx_stat_discard                   : 0
rx_stat_err                       : 0
rx_tagged_frames                  : 0
rx_total_frames                   : 250
rx_trunc_frames                   : 0
rx_ucast_frames                   : 0
rx_undrsz_frames                  : 0
rx_unsupported_da_pausepfc_frames : 0
rx_unsupported_opcode_frames      : 0
rx_wrong_sa_frames                : 0
tx_1024B_1518_frames              : 0
tx_128B_255B_frames               : 98
tx_1519B_2047_frames              : 0
tx_2048B_4095B_frames             : 0
tx_256B_511B_frames               : 36
tx_4096B_9216B_frames             : 0
tx_512B_1023B_frames              : 0
tx_64B_frames                     : 18
tx_65B_127B_frames                : 68
tx_9217B_16383B_frames            : 0
tx_bcast_frames                   : 56
tx_bytes                          : 39599
tx_cnp_bytes                      : 108847364832208
tx_cnp_pkts                       : 0
tx_control_frames                 : 0
tx_dbl_tagged_frames              : 0
tx_eee_lpi_duration               : 0
tx_eee_lpi_events                 : 0
```

```
tx_err                          : 0
tx_excessive_coll_frames        : 0
tx_fcs_err_frames               : 0
tx_fifo_underruns               : 0
tx_frag_frames                  : 0
tx_good_frames                  : 220
tx_good_vlan_frames             : 0
tx_hcfc_msgs                    : 0
tx_jabber_frames                : 0
tx_late_coll_frames             : 0
tx_llfc_logical_msgs            : 0
tx_mcast_frames                 : 164
tx_multi_coll_frames            : 0
tx_multi_dfrl_frames            : 0
tx_oversz_frames                : 0
tx_pause_frames                 : 0
tx_pfc_ena_frames_pri0          : 0
tx_pfc_ena_frames_pri1          : 0
tx_pfc_ena_frames_pri2          : 0
tx_pfc_ena_frames_pri3          : 0
tx_pfc_ena_frames_pri4          : 0
tx_pfc_ena_frames_pri5          : 0
tx_pfc_ena_frames_pri6          : 0
tx_pfc_ena_frames_pri7          : 0
tx_pfc_frames                   : 0
tx_rocev1_bytes                 : 108847364832208
tx_rocev1_pkts                  : 0
tx_rocev2_bytes                 : 108847364832208
tx_rocev2_pkts                  : 0
tx_runt_frames                  : 0
tx_single_coll_frames           : 0
tx_single_dfrl_frames           : 0
tx_stat_discard                 : 0
tx_stat_error                   : 0
tx_tagged_frames                : 0
tx_total_collisions             : 0
tx_total_frames                 : 220
tx_ucast_frames                 : 0
tx_xthol_frames                 : 0
PSComputerName                  : XXXXX
```

Performance Counters for Windows

Provides the counter sets that are exposed by the miniport driver. Currently, three counter sets are exposed.

Windows Operating System

The Windows operating system provides an application, `perfmon.exe` , that allows counters and statistics from various sources to be displayed and collected. Components that expose performance counters are called performance counter providers. After adding custom performance counter support, the Ethernet Network Adapter driver is now a performance

counter provider. This means that the driver makes a number of counters visible via the `perfmon.exe` application and via `powershell` .

Counter Sets

Performance counters are contained within counter sets. Each counter set is a container for a group of counters. This is done for organizational reasons, and for the user to more easily locate counters of interest.

Manifest File

All kernel mode performance counter providers must have a manifest file (.man extension) that describes the counter sets and the counters within those sets. The manifest file defines a GUID for each counter set. The manifest file also contains the name of each counter set, name of each counter within the counter set, and the size (u32 or u64) of each counter. This manifest file is used for two reasons:

- It is a source file component for the driver and is required to compile the driver.
- It is an installable component that must accompany the driver `.sys` file.

Installing the Manifest File

When a device driver that is a performance counter provider is installed, the performance counters that it exposes do not automatically show up in `perfmon.exe` . In order for the counters to be visible in `perfmon.exe` , the manifest file must be installed after the driver is installed. If the manifest file is not installed, the driver still functions correctly, however performance counters are not available. The installation of the manifest file is performed by an operating system tool called `lodctr.exe` . This executable is included with Windows. There is no way to install the manifest file using the driver installation INF file.

Installing with a New Driver

When installing on an operating system that does not have a previous version of the Ethernet Network Adapter driver, or is using an inbox driver, the following steps must be performed:

1. Install or update to the new driver that supports performance counters and ensure that the new driver has started.
2. Copy the manifest file to some location on the target system such as `c:\temp`.
3. Ensure that a copy of the new `.sys` file that supports performance counters is in the same folder as the manifest file. This is a copy of the `.sys` file that is running from step 1.
4. From an administrator command prompt, run the following command:

```
lodctr.exe /m:manifest_file.man
```

The performance counters are visible in `perfmon.exe` . The manifest file and the `.sys` file which were in the temporary location (such as `c:\temp`) are not required for runtime operation and may be removed. The manifest file is stored so that the performance counters may be uninstalled in the future.

Updating the Driver

When an existing driver that supports performance counters is being removed or updated to a newer version, the currently installed manifest file must be removed and a new manifest must be installed. If this is not done, then counter sets may show invalid values if the counter set has changed in the new driver version, or may show “unavailable”. To update to a new driver, use the following steps:

1. Uninstall the existing manifest. The manifest file for the currently installed driver is required to perform this step. From an administrator command prompt, run the following command:

```
unlodctr.exe /m:manifest_file.man.
```

2. Install or update to the new driver that supports performance counters and ensure that the new driver has started.
3. Copy the manifest file to some location on the target system such as `c:\temp` .
4. Ensure that a copy of the new `.sys` file that supports performance counters is in the same folder as the manifest file. This is a copy of the `.sys` file that is running from step 1.

5. From an administrator command prompt, run the following command:

```
lodctr.exe /m:manifest_file.man
```

If the `.sys` file is not present in the same folder as the manifest file when `lodctr.exe` is performed, no error occurs. However, the performance counters will not be visible.

Uninstalling the Manifest File

Uninstalling manifest is performed using the command `unlodctr.exe /m:manifest_file.man` . Unlike installation, the `.sys` file does not need to be present in the same folder as the manifest file, and the driver does not need to be loaded at the time the manifest is uninstalled. Uninstalling the manifest file removes the custom performance counters from `perfmon.exe` . However, it does not uninstall the driver or remove any functionality. Uninstalling the manifest file should be performed using the same manifest (`.man` file) that was used to install the manifest. Newer manifest files from updated drivers cannot be used to uninstall older manifests. This is because the counter sets are uninstalled using their associated GUIDs and newer manifest files may have modified counter sets with new GUIDs.

RDMA Resources Counter Set

This counter displays the number of currently active RDMA resources. The resource types tracked are listed as follows:

- **PD** – Protection Domains currently in use. PDs are container objects used to isolate QPs and MRs. They can be created by kernel mode or user more RDMA clients. Each user mode client that performs an OpenAdapter call will cause a new PD to be created.
- **QP** – RDMA Queue Pairs currently in use. QPs are created by kernel mode or user more RDMA clients.
- **CQ** – RDMA Completion Queues currently in use. CQs are created by kernel mode or user more RDMA clients.
- **SRQ** – Shared Receive Queues currently in use. SRQs are created by kernel mode or user more RDMA clients.
- **MR** – Memory Regions currently in use. Includes conventional and fast register type MRs. MRs are created by kernel mode or user more RDMA clients.
- **MW** – Memory Windows currently in use. MWs are created by kernel mode or user more RDMA clients.
- **LAM** – Logical Address Mappings currently in use. LAMs are sets of locked pages. Pages within the set can be used for fast memory registration operations on a QPs SQ. LAMs are only used by kernel mode RDMA consumers.
- **AH** – Address Handles currently in use. Address handles represent an active address (to a peer) in an address handle table. AHs are created by kernel mode or user more RDMA clients.
- **NDSPI Clients** – The current number of user mode RDMA clients. This value increments each time an NDSPI client performs an `OpenAdapter()` call to the NDSPI provider DLL.

RDMA Kernel Allocations Counter Set

This counter set tracks resources, similar to the RDMA resource counter set, however, it shows an allocation and free count for each resource. This can give the viewer an idea of how frequently a resource is being used and released. This counter set shows allocations and frees that occur in kernel mode.

- **QP Allocated/Freed** – Total number of kernel mode QP allocations and frees. The difference between the allocated and free values is the number of kernel QPs currently in use.
- **CQ Allocated/Freed** – Total number of kernel mode CQ allocations and frees. The difference between the allocated and free values is the number of kernel CQs currently in use.
- **SRQ Allocated/Freed** – Total number of kernel mode SRQ allocations and frees. The difference between the allocated and free values is the number of kernel SRQs currently in use.
- **MR Allocated/Freed** – Total number of kernel mode MR allocations and frees. The difference between the allocated and free values is the number of kernel MRs currently in use.
- **MW Allocated/Freed** – Total number of kernel mode MW allocations and frees. The difference between the allocated and free values is the number of kernel MWs currently in use.

- **LAM Allocated/Freed** – Total number of LAM allocations and frees. The difference between the allocated and free values is the number of LAMs currently in use. LAMs are only allocated and freed by kernel mode RDMA clients.

RDMA User Allocations Counter Set

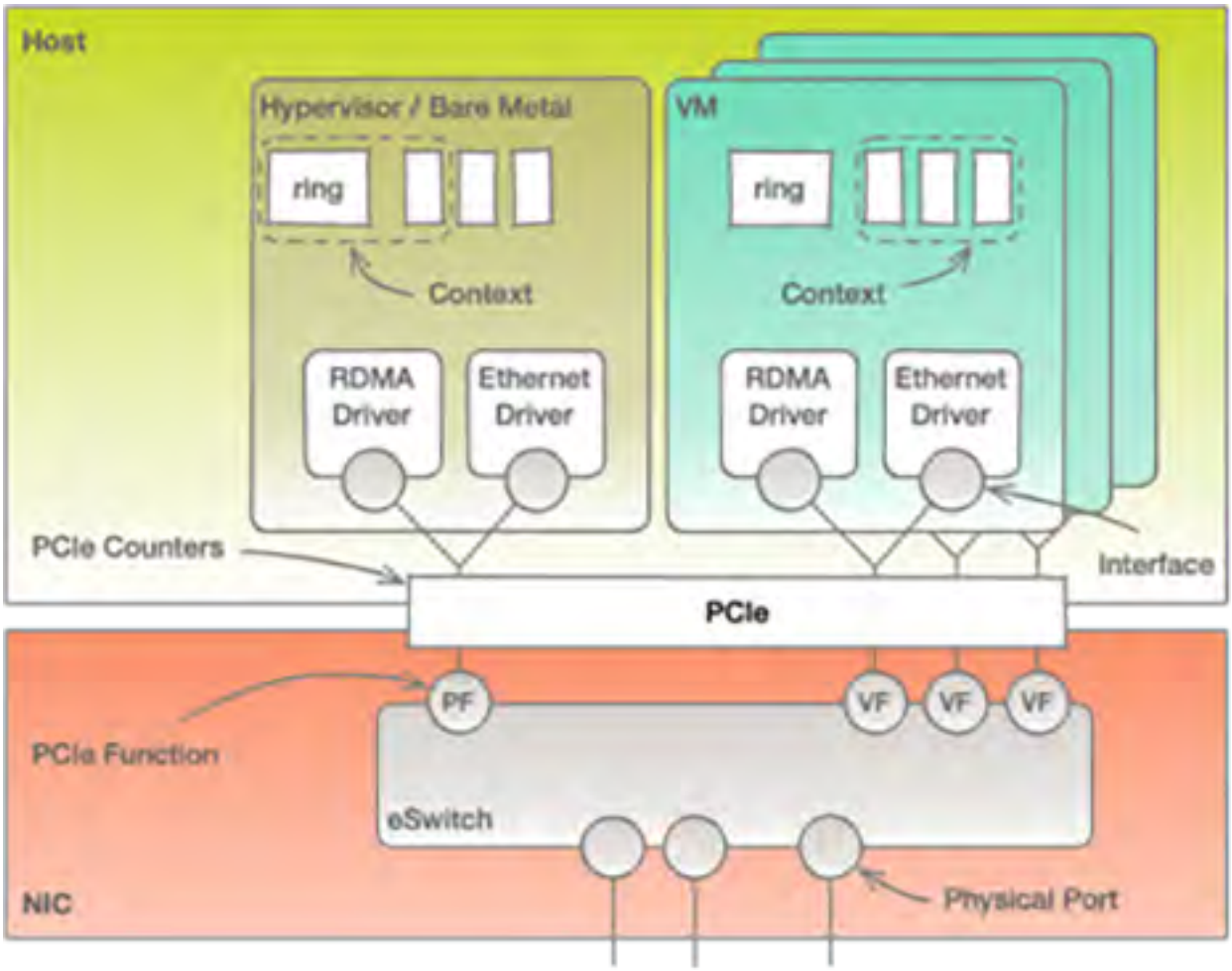
This counter set tracks resources, similar to the RDMA resource counter set; however, it shows an allocation and free count for each resource. This can give the viewer an idea of how frequently a resource is being used and released. This counter set shows allocations and frees that occur in user mode via the NDSPI DLL.

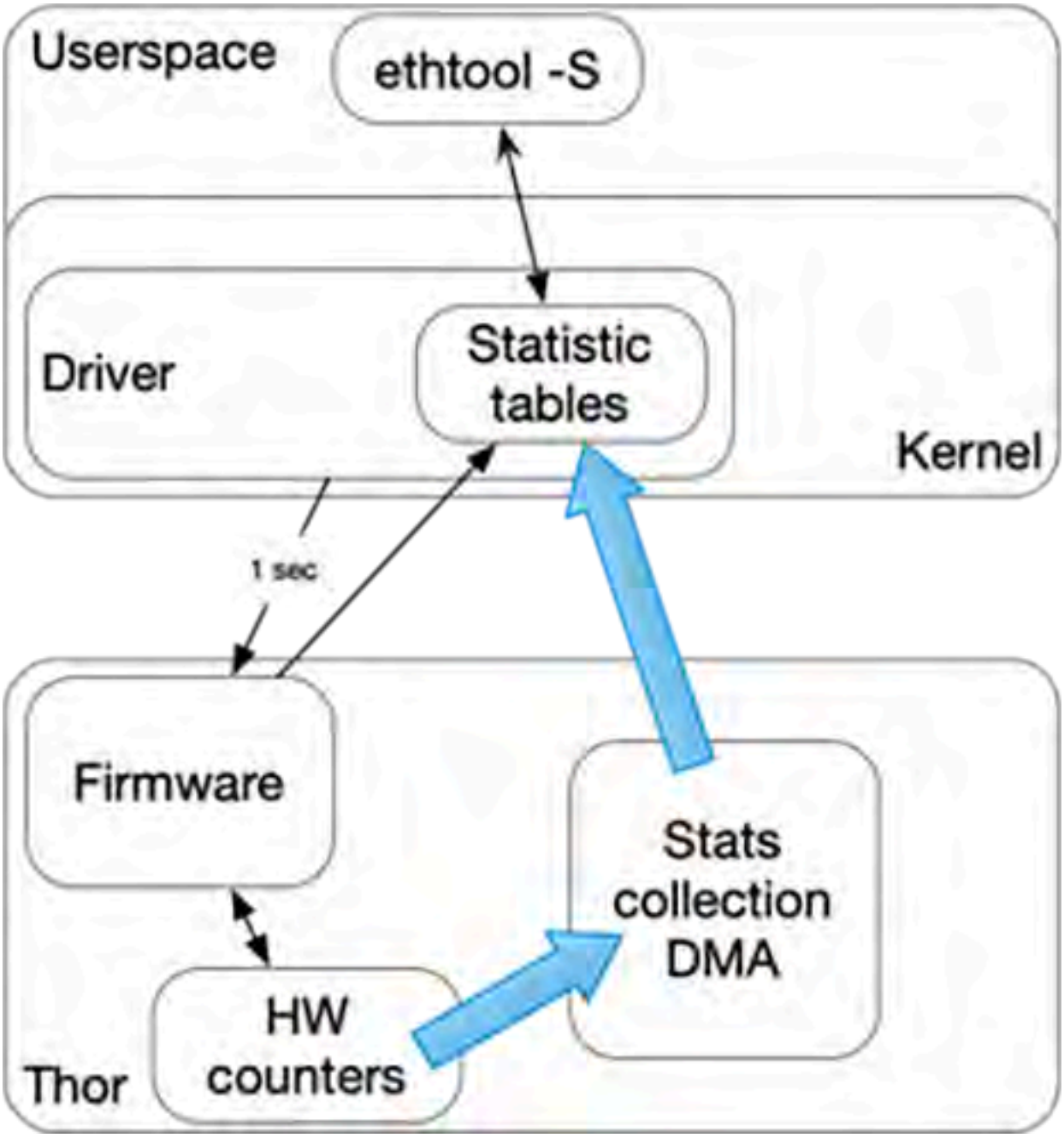
- **QP Allocated/Freed** – Total number of user mode QP allocations and frees. The difference between the allocated and free values is the number of user QPs currently in use.
- **CQ Allocated/Freed** – Total number of user mode CQ allocations and frees. The difference between the allocated and free values is the number of user CQs currently in use.
- **SRQ Allocated/Freed** – Total number of user mode SRQ allocations and frees. The difference between the allocated and free values is the number of user SRQs currently in use.
- **MR Allocated/Freed** – Total number of user mode MR allocations and frees. The difference between the allocated and free values is the number of user MRs currently in use.
- **MW Allocated/Freed** – Total number of user mode MW allocations and frees. The difference between the allocated and free values is the number of user MWs currently in use.

Ethtool Counters

Provides information on the available ethtool counters applicable to all Broadcom network adapters.

Several counter groups depend on where the counter is being counted. In addition, each group of counters may have a different counter type. See the following figures for additional information.


Host
Hypervisor / Bare Metal
ring
Context
RDMA Driver
Ethernet Driver
VM
ring
Context
RDMA Driver
Ethernet Driver
PCIe Counters
Interface
PCIe
PF
VF
VF
VF
PCIe Function
eSwitch
NIC
Physical Port



Counter Groups

The following counter group types are supported:

- Physical Port Counters
- Ring Counters
- Ifconfig counters
- Extended Port Stats

Counter Types

The following table lists the available counter types.

Physical Port Counters

The physical port counters are described in the following table.

Table 43: Physical Port Counters

Physical Port Counters	Description	Type	Available
rx_64b_frames	Number of 64 Bytes frames received on a port	port	All Devices
pcie_non_posted_packet_count	Displays the count of nonposted packets sent to the PCIe.	port	All Devices
rx_65b_127b_frames	Number of 65-127 Bytes frames received on a port	port	All Devices
rx_128b_255b_frames	Number of 128-255 Bytes frames received on a port	port	All Devices
rx_256b_511b_frames	Number of 256-511 Bytes frames received on a port	port	All Devices
rx_512b_1023b_frames	Number of 512-1023 Bytes frames received on a port	port	All Devices
rx_1024b_1518b_frames	Number of 1024-1518 Bytes frames received on a port	port	All Devices
rx_good_vlan_frames	Number of each good VLAN (excludes FCS errors) frame received which is 1519 to 1522 bytes in length inclusive (excluding framing bits but including FCS bytes) on a port	port	All Devices
rx_1519b_2047b_frames	Number of 1519-2047 Bytes frames received on a port	port	All Devices
rx_2048b_4095b_frames	Number of 2048-4095 Bytes frames received on a port	port	All Devices
rx_4096b_9216b_frames	Number of 4096-9216 Bytes frames received on a port	port	All Devices
rx_9217b_16383b_frames	Number of 9217-16383 Bytes frames received on a port	port	All Devices
rx_total_frames	Number of frames received on a port	port	All Devices
rx_ucast_frames	Number of unicast frames received on a port	port	All Devices
rx_mcast_frames	Number of multicast frames received on a port	port	All Devices
rx_bcast_frames	Number of broadcast frames received on a port	port	All Devices

Physical Port Counters	Description	Type	Available
rx_fcs_err_frames	Number of received frames with FCS error on a port	port	All Devices
rx_ctrl_frames	Number of control frames received on a port	port	All Devices
rx_pause_frames	Number of PAUSE frames received on a port	port	All Devices
rx_pfc_frames	Number of PFC frames received on a port	port	All Devices
rx_align_err_frames	Number of received packets with alignment error on a port	port	All Devices
rx_ovrsz_frames	Number of oversized frames received on a port. The counter will increment if the packet size > MTU and <= 9626B in single function mode. The counter is disabled in any multi-function mode (multi-root/multi-host).	port	All Devices
rx_jbr_frames	Number of jabber packets received on a port	port	All Devices
rx_mtu_err_frames	Number of received frames with MTU error on a port. The counter increments if packet size > 9626B"	port	All Devices
rx_tagged_frames	Number of received frames with one or two VLAN tags on a port	port	All Devices
rx_double_tagged_frames	Number of received frames with two VLAN tags on a port	port	All Devices
rx_good_frames	Number of good frames (without errors) received on a port	port	All Devices
rx_pfc_ena_frames_pri0	Number of received PFC frames with PFC enabled bit for Pri 0 on a port	port	All Devices
rx_pfc_ena_frames_pri1	Number of received PFC frames with PFC enabled bit for Pri 1 on a port	port	All Devices
rx_pfc_ena_frames_pri2	Number of received PFC frames with PFC enabled bit for Pri 2 on a port	port	All Devices
rx_pfc_ena_frames_pri3	Number of received PFC frames with PFC enabled bit for Pri 3 on a port	port	All Devices
rx_pfc_ena_frames_pri4	Number of received PFC frames with PFC enabled bit for Pri 4 on a port	port	All Devices

Physical Port Counters	Description	Type	Available
rx_pfc_ena_frames_pri5	Number of received PFC frames with PFC enabled bit for Pri 5 on a port	port	All Devices
rx_pfc_ena_frames_pri6	Number of received PFC frames with PFC enabled bit for Pri 6 on a port	port	All Devices
rx_pfc_ena_frames_pri7	Number of received PFC frames with PFC enabled bit for Pri 7 on a port	port	All Devices
rx_undrsz_frames	Number of undersized frames received on a port	port	All Devices
rx_eee_lpi_events	Number of RX EEE LPI Events on a port	port	All Devices
rx_eee_lpi_duration	EEE LPI Duration Counter on RX on a port	port	All Devices
rx_bytes	Number of received bytes on a port	port	All Devices
rx_runt_bytes	Number of bytes received in runt frames on a port	port	All Devices
rx_runt_frames	Number of runt frames received on a port	port	All Devices
rx_stat_discard	Number of packets discarded at the network port due to lack of network side buffer resources	port	All Devices
rx_stat_err	Number of packets dropped at the network port due to errors (including the errors at the MAC/PHY level)	port	All Devices
tx_64b_frames	Number of 64 Bytes frames transmitted on a port	port	All Devices
tx_65b_127b_frames	Number of 65-127 Bytes frames transmitted on a port	port	All Devices
tx_128b_255b_frames	Number of 128-255 Bytes frames transmitted on a port	port	All Devices
tx_1518b_frames	Number of 256-511 Bytes frames transmitted on a port	port	All Devices
tx_512b_1023b_frames	Number of 512-1023 Bytes frames transmitted on a port	port	All Devices
tx_1024b_1518_frames	Number of 1024-1518 Bytes frames transmitted on a port	port	All Devices

Physical Port Counters	Description	Type	Available
tx_good_vlan_frames	Number of each good VLAN (excludes FCS errors) frame transmitted which is 1519 to 1522 bytes in length inclusive (excluding framing bits but including FCS bytes) on a port	port	All Devices
tx_1519b_2047_frames	Number of 1519-2047 Bytes frames transmitted on a port	port	All Devices
tx_2048b_4095b_frames	Number of 2048-4095 Bytes frames transmitted on a port	port	All Devices
tx_4096b_9216b_frames	Number of 4096-9216 Bytes frames transmitted on a port	port	All Devices
tx_9217b_16383b_frames	Number of 9217-16383 Bytes frames transmitted on a port	port	All Devices
tx_good_frames	Number of good frames transmitted on a port	port	All Devices
tx_total_frames	Number of frames transmitted on a port	port	All Devices
tx_ucast_frames	Number of unicast frames transmitted on a port	port	All Devices
tx_mcast_frames	Number of multicast frames transmitted on a port	port	All Devices
tx_bcast_frames	Number of broadcast frames transmitted on a port	port	All Devices
tx_pause_frames	Number of PAUSE control frames transmitted on a port	port	All Devices
tx_jabber_frames	Number of jabber frames transmitted on a port	port	All Devices
tx_fcs_err_frames	Number of frames transmitted with FCS error on a port	port	All Devices
tx_err	Number of transmit errors on a port	port	All Devices
tx_fifo_underruns	Number of TX FIFO underruns on a port	port	All Devices
tx_pfc_ena_frames_pri0	Number of PFC frames with PFC enabled bit for Pri 0 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri1	Number of PFC frames with PFC enabled bit for Pri 1 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri2	Number of PFC frames with PFC enabled bit for Pri 2 transmitted on a port	port	All Devices

Physical Port Counters	Description	Type	Available
tx_pfc_ena_frames_pri3	Number of PFC frames with PFC enabled bit for Pri 3 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri4	Number of PFC frames with PFC enabled bit for Pri 4 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri5	Number of PFC frames with PFC enabled bit for Pri 5 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri6	Number of PFC frames with PFC enabled bit for Pri 5 transmitted on a port	port	All Devices
tx_pfc_ena_frames_pri7	Number of PFC frames with PFC enabled bit for Pri 7 transmitted on a port	port	All Devices
tx_eee_lpi_events	Number of EEE LPI Events on TX on a port	port	All Devices
tx_eee_lpi_duration	EEE LPI Duration Counter on TX on a port	port	All Devices
tx_total_collisions	Number of TX collisions on a port	port	All Devices
tx_bytes	Number of transmitted bytes on a port	port	All Devices
tx_xthol_frames	Number of end-to-end HOL frames	port	All Devices
tx_stat_discard	Number of transmit packets discarded at the network port due to insufficient network side buffer resources	port	All Devices
tx_stat_error	Number of packets dropped at the network port due to errors (including the errors at the MAC/PHY level)	port	All Devices
pcie_statistics_tx_tlp	Number of TLP bytes that have been transmitted for the caller PF.	port	BCM95750X/BCM957608
pcie_statistics_rx_tlp	Number of TLP bytes that have been received for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_hdr_posted	Posted Header Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_hdr_nonposted	Non-posted Header Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608

Physical Port Counters	Description	Type	Available
pcie_credit_fc_hdr_cmpl	Completion Header Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_data_posted	Posted Data Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_data_nonposted	Non-Posted Data Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_data_cmpl	Completion Data Flow Control credits available for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_tgt_nonposted	Available Non-posted credit for target flow control reads or config for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_tgt_data_posted	Available posted data credit for target flow control writes for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_tgt_hdr_posted	Available posted header credit for target flow control writes for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_cmpl_hdr_posted	Available completion flow control header credits for the caller PF.	port	BCM95750X/BCM957608
pcie_credit_fc_cmpl_data_posted	Available completion flow control data credits.	port	BCM95750X/BCM957608
pcie_cmpl_longest	Displays Time information of the longest completion time from any of the 4 tags for the caller PF. The unit of time recorded is in microseconds.	port	BCM95750X/BCM957608
pcie_cmpl_shortest	Displays Time information of the shortest completion time from any of the 4 tags for the caller PF. The unit of time recorded is in microseconds.	port	BCM95750X/BCM957608
pcie_non_posted_packet_count	Displays the count of nonposted packets sent to the PCIe.	port	All Devices
pcie_other_packet_count	Displays the count of other packets (for example, not posted or non-posted) sent to the PCIe.	port	All Devices

Physical Port Counters	Description	Type	Available
pcie_tl_credit_nph_histogram	Displays the TL credit histogram counter buckets for non-postedheaders. This field represents an array of 8 buckets where each bucket stores the counters in the following credit range: • NPH Credit = 0 :0x0 • NPH Credit = 1 :0x0 • NPH Credit = 2 to 3 :0x0 • NPH Credit = 4 to 7 :0x0 • NPH Credit = 8 to 15 :0x0 • NPH Credit =16 to 3 :0x0 • NPH Credit = 32 to 63 :0x0 • NPH Credit = 64 and higher:0x35350d5e	port	All Devices
pcie_tl_credit_pd_histogram	Displays the TL credit histogram counter buckets for posteddata. This field represents an array of 8 buckets where each bucket stores the counters in the following credit range: • PD Credit = 0 :0x0 • PD Credit = 1 to 3 :0x0 • PD Credit = 4 to 7 :0x0 • PD Credit = 8 to 15 :0x0 • PD Credit = 16 to 31 :0x0 • PD Credit = 32 to 63 :0x0 • PD Credit = 64 to 127 :0x0 • PD Credit = 128 and higher :0x35350d5e	port	All Devices
pcie_tl_credit_ph_histogram	Displays the TL credit histogram counter buckets for postedheaders. This field represents an array of buckets where each bucket stores the counters in the following credit range: • PH Credit = 0 :0x0 • PH Credit = 1 :0x0 • PH Credit = 2 to 3 :0x0 • PH Credit = 4 to 7 :0x0 • PH Credit = 8 to 15 :0x0 • PH Credit = 16 to 31 :0x0 • PH Credit = 32 to 63 :0x0 • PH Credit = 64 and higher :0x35350d5e	port	All Devices

Physical Port Counters	Description	Type	Available
pcie_wr_latency_histogram	Displays the PCIe write latency histogram data counter values. This field represents an array of 12. The first 8 entries represent the bucket counters each bucket counts the latency of the following range in clock cycles: • 0 = 0 - 3 • 1 = 32 - 63 • 2 = 64 - 95 • 3 = 96 - 127 • 4 = 128 - 159 • 5 = 160 - 191 • 6 = 192 - 223 • 7 = 224 and higher The last 4 entries contains the below data: • 8 = MIN data value • 9 = MAX data value • 10 = Event counter	port	All Devices
cache_miss_count_cfcq	Contains the total number of CFCQ 'misses' observed for all the PFs.	port	BCM95750X/BCM957608
cache_miss_count_cfcs	Contains the total number of CFCS 'misses' observed for all the PFs.	port	BCM95750X/BCM957608
cache_miss_count_cfcc	Contains the total number of CFCC 'misses' observed for all the PFs.	port	BCM95750X/BCM957608
cache_miss_count_cfcm	Contains the total number of CFCM 'misses' observed for all the PFs.	port	BCM95750X/BCM957608

Ring Counters

The ring counters are described in the following table.

Table 44: Ring Counters

Ring Counters	Description	Type	Available
rx_hds_pkt	–	ring	All Devices
rx_tpa_hds_pkt	–	ring	All Devices
rx_resets	Number of resets on RX ring i	ring	All Devices
rx_ucast_packets	Number of received unicast packets on ring i	ring	All Devices
rx_mcast_packets	Number of received multicast packets on ring i	ring	All Devices

Ring Counters	Description	Type	Available
rx_bcast_packets	Number of received broadcast packets on ring i	ring	All Devices
rx_discards	Number of received packets discarded on ring i due to lack of host side ring buffer resources	ring	All Devices
rx_errors	Number of received packets dropped on ring i due to errors or flow processing policies. **Note:** In older kernels, this counter is named as rx_drops.	ring	All Devices
rx_ucast_bytes	Number of received bytes for unicast traffic on ring i	ring	All Devices
rx_mcast_bytes	Number of received bytes for multicast traffic on ring i	ring	All Devices
rx_bcast_bytes	Number of received bytes for broadcast traffic on ring i	ring	All Devices
tx_ucast_packets	Number of transmitted unicast packets on ring i	ring	All Devices
tx_mcast_packets	Number of transmitted multicast packets on ring i	ring	All Devices
tx_bcast_packets	Number of transmitted broadcast packets on ring i	ring	All Devices
tx_errors	Number of transmit packets dropped on ring i due to reasons including malicious VF behavior, packet being larger than MTU, or having an invalid/spoofed MAC address or VLAN field. **Note:** In older kernels, this counter is named as tx_drops.	ring	All Devices
tx_discards	Number of transmit packets discarded on ring i. This counter should always be 0	ring	All Devices
tx_ucast_bytes	Number of transmitted bytes for unicast traffic on ring i	ring	All Devices
tx_mcast_bytes	Number of transmitted bytes for multicast traffic on ring i	ring	All Devices
tx_bcast_bytes	Number of transmitted bytes for broadcast traffic on ring i	ring	All Devices
tpa_packets	Number of aggregated packets on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95741X

Ring Counters	Description	Type	Available
tpa_bytes	Number of TCP payload bytes of aggregated packets on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95741X
tpa_events	Number of packet aggregation events on ring i. Aggregation is based on a mechanism like Large Receive Offload (LRO)	ring	BCM95741X
tpa_aborts	Number of aborted packet aggregation on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95741X
rx_l4_csum_errors	Number of packets received with invalid L4 on ring i	ring	All Devices
rx_tpa_eligible_pkt	Number of packets eligible for aggregation on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95750X/BCM957608
rx_tpa_eligible_bytes	Number of bytes of packets eligible for aggregation on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95750X/BCM957608
rx_tpa_pkt	Number of aggregated packets on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95750X/BCM957608
rx_hds_pkt	Number of received packets on ring i with L4 payload split into page buffers.	ring	All devices
rx_tpa_hds_pkt	Number of received aggregated TCP packets on ring i with TCP payload split into page buffers	ring	All devices
rx_tpa_bytes	Number of TCP payload bytes of aggregated packets on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95750X/BCM957608
rx_tpa_errors	Number of aggregation errors on ring i. Packet aggregation is based on a mechanism. e.g. Large Receive Offload (LRO)	ring	BCM95750X/BCM957608
rx_tpa_events:		ring	BCM957608

Ring Counters	Description	Type	Available
so_txtime_xmit		ring	BCM957608
so_txtime_cmpl_errors		ring	BCM957608
rx_buf_errors	Number of buffer errors on ring i. This statistic is maintained by the driver	ring	All Devices
missed_irqs	Number of missed IRQ on ring i. This statistic is maintained by the driver	ring	All Devices
rx_total_oom_discard	Total number of packets dropped due to buffer or SKB allocation errors	ring	All Devices
rx_total_netpoll_discards	Total number of packets dropped because of netpoll	ring	All Devices
rx_resets	Number of resets on RX ring i	ring	All Devices
rx_total_l4_csum_errors	Total number of packets received with invalid L4 across all rings	ring	All Devices
rx_total_resets	Total number of resets across all Rx rings	ring	All Devices
rx_total_buf_errors	Total number of buffer errors across all rings	ring	All Devices
tx_total_resets	Total number of resets across all Tx rings	ring	All Devices
tx_total_ring_discards	Total number of transmitted packets discarded across all rings	ring	All Devices
total_missed_irqs	Total number of missed irqs across all rings	ring	All Devices

Host Counters

The host counters are the sum of all ring counters on a host. They are shown in the following table.

Table 45: Host Counters

Ifconfig Counters	Description	Type	Available
RX packets	Number of packets received via the interface on a host	port	All Devices
RX bytes	Number of bytes received on a host	port	All Devices
RX errors	Number of damaged packets received on a host	port	All Devices
RX dropped	Number of dropped packets due to reception errors on a host	port	All Devices

Ifconfig Counters	Description	Type	Available
RX overruns	Number of received packets that experienced data overruns on a host	port	All Devices
RX frame	Number of received packets that experienced frame errors on a host	port	All Devices
TX packets	Number of packets transmitted via the interface on a host	port	All Devices
TX bytes	Number of bytes transmitted on a host	port	All Devices
TX errors	Number of packets that experienced transmission error on a host	port	All Devices
TX dropped	Number of dropped transmitted packets due to transmission errors on a host	port	All Devices
TX overruns	Number of transmitted packets that experienced data overruns on a host	port	All Devices
TX carrier	Number of received packets that experienced loss of carriers per host	port	All Devices
TX collision	Number of transmitted packets that experienced Ethernet collisions on a host. A non-zero value of this field indicates possibility of network congestion	port	All Devices

Extended Port Status

The extended port stats are shown in the following table.

Table 46: Extended Port Stats

Extended Port Stats	Description	Type	Available
link_down_events	Number of times link state transitioned to down	port	BCM95750X/BCM957608
continuous_pause_events	Number of times the idle rings with pause bit are found	port	BCM95750X/BCM957608
resume_pause_events	Number of times the active rings pause bit resumed back	port	BCM95750X/BCM957608
continuous_roce_pause_events	Number of times, the ROCE cos queue PFC is disabled to avoid pause flood/burst	port	BCM95750X/BCM957608
resume_roce_pause_events	Number of times, the ROCE cos queue PFC is enabled back	port	BCM95750X/BCM957608

Extended Port Stats	Description	Type	Available
rx_bytes_cos[i]	Number of bytes received on the specific CoS i (bytes from dropped frames not included)	port	BCM95750X/BCM957608
rx_packets_cos[i]	Number frames received on the specific CoS i (dropped frames not included)	port	BCM95750X/BCM957608
pfc_pri[i]_rx_duration_us	Time duration receiving a XON -> XOFF and a subsequent XOFF -> XON for priority i	port	BCM95750X/BCM957608
pfc_pri[i]_rx_transitions	Number of times, a XON -> XOFF and XOFF -> XON transitions occur for priority i	port	BCM95750X/BCM957608
rx_bits	Number of received bits	port	BCM95750X/BCM957608
rx_buffer_passed_threshold	Number of events where the port receive buffer was over 85% full	port	BCM95750X/BCM957608
rx_pcs_symbol_err	Number of symbol errors that were not corrected by FEC correction algorithm	port	BCM95750X/BCM957608
rx_corrected_bits	Number of corrected bits on the port according to active FEC	port	BCM95750X/BCM957608
rx_discard_bytes_cos[i]	Number bytes dropped on the specific CoS i	port	BCM95750X/BCM957608
rx_discard_packets_cos[i]	Number frames discarded on the specific CoS i.	port	BCM95750X/BCM957608
tx_bytes_cos[i]	Number bytes transmitted on the specific CoS i (bytes from dropped frames not included)	port	BCM95750X/BCM957608
tx_packets_cos[i]	Number frames transmitted on the specific CoS i (dropped frames not included)	port	BCM95750X/BCM957608
pfc_pri[i]_tx_duration_us	Time duration between transmitting a XON -> XOFF and a subsequent XOFF -> XON for priority i	port	BCM95750X/BCM957608
pfc_pri[i]_tx_transitions	Number of times, a XON -> XOFF and XOFF -> XON transitions occur for priority i	port	BCM95750X/BCM957608
rx_bytes_pri[i]	Number bytes received with PRI set to i (bytes from dropped frames not included)	port	BCM95750X/BCM957608

Extended Port Stats	Description	Type	Available
rx_packets_pri[i]	Number packets received with PRI set to i (dropped frames not included)	port	BCM95750X/BCM957608
tx_bytes_pri[i]	Number bytes transmitted with PRI set to i (bytes from dropped frames not included)	port	BCM95750X/BCM957608
tx_packets_pri[i]	Number packets transmitted with PRI set to i (dropped frames not included)	port	BCM95750X/BCM957608
rx_fec_corrected_blocks	Counts the number of FEC blocks corrected by the FEC function	port	BCM95750X/BCM957608
rx_fec_uncorrectable_blocks	Counts the number of FEC blocks that are determined to be uncorrectable by the FEC function	port	BCM95750X/BCM957608
rx_filter_miss	Total number of packets that are dropped due to not matching any RX filter rules	port	BCM957608

Loopback Port Statistics

The loopback port statistics are shown in the following table.

Table 47: Loopback Port Statistics

Extended Port Stats	Description	Type	Available
lpbk_ucast_frames	Number of transmitted unicast frames.	uint64	BCM957608
lpbk_mcast_frames	Number of transmitted multicast frames.	uint64	BCM957608
lpbk_bcast_frames	Number of transmitted broadcast frames.	uint64	BCM957608
lpbk_ucast_bytes	Number of transmitted bytes for unicast traffic.	uint64	BCM957608
lpbk_mcast_bytes	Number of transmitted bytes for multicast traffic.	uint64	BCM957608
lpbk_bcast_bytes	Number of transmitted bytes for broadcast traffic.	uint64	BCM957608
lpbk_tx_discards	Number of dropped TX packets.	uint64	BCM957608
lpbk_tx_errors	Number of error dropped RX packets.	uint64	BCM957608
lpbk_rx_discards	Number of dropped RX packets.	uint64	BCM957608
lpbk_rx_errors	Number of error dropped RX packets.	uint64	BCM957608

Hardware Doorbell Recovery Counters

The hardware doorbell recovery counters are shown in the following table.

Note: These statistics are available per device only.

Table 48: Hardware Doorbell Recovery Counters

Extended Port Stats	Description	Type	Available
hw_db_recov_dbs_dropped	Total number of Doorbell messages dropped from the DB FIFO. This counter is only applicable for devices that support the hardware based doorbell drop recovery feature.	uint64	BCM95750X/BCM957608
hw_db_recov_drops_serviced	Total number of doorbell drops serviced. This counter is only applicable for devices that support the hardware based doorbell drop recovery feature.	uint64	BCM95750X/BCM957608
hw_db_recov_dbs_recovered	Total number of dropped doorbells recovered. This counter is only applicable for devices that support the hardware based doorbell drop recovery feature.	uint64	BCM95750X/BCM957608
hw_db_recov_oo_drop_count	Total number of out of order doorbell messages dropped. This counter is only applicable for devices that support the hardware based doorbell drop recovery feature.	uint64	BCM95750X/BCM957608

COS Statistics Counters

The COS statistics counters are shown in the following table.

Table 49: COS Statistics Counters

Extended Port Stats	Description	Type	Available
rx_bytes_cos[i]	Number of bytes received on the specific CoS i (bytes from dropped frames not included).	port	BCM95750X/BCM957608
rx_packets_cos[i]	Number frames received on the specific CoS i (dropped frames not included).	port	BCM95750X/BCM957608
rx_discard_bytes_cos[i]	Number bytes dropped on the specific CoS i.	port	All Devices
rx_discard_packets_cos[i]	Number frames discarded on the specific CoS i.	port	All Devices

Extended Port Stats	Description	Type	Available
tx_bytes_cos[i]	Number bytes transmitted on the specific CoS i (bytes from dropped frames not included).	port	All Devices
tx_packets_cos[i]	Number frames transmitted on the specific CoS i (dropped frames not included).	port	All Devices

PFC Counters

The PFC counters are shown in the following table.

Table 50: PFC Counters

Extended Port Stats	Description	Type	Available
rx_pfc_frames	Number of PFC frames received on a port.	port	All Devices
rx_pfc_ena_frames_pri0	Number of received PFC frames with PFC enabled bit for Pri 0 on a port.	port	All Devices
rx_pfc_ena_frames_pri1	Number of received PFC frames with PFC enabled bit for Pri 1 on a port.	port	All Devices
rx_pfc_ena_frames_pri2	Number of received PFC frames with PFC enabled bit for Pri 2 on a port.	port	All Devices
rx_pfc_ena_frames_pri3	Number of received PFC frames with PFC enabled bit for Pri 3 on a port.	port	All Devices
rx_pfc_ena_frames_pri4	Number of received PFC frames with PFC enabled bit for Pri 4 on a port.	port	All Devices
x_pfc_ena_frames_pri5	Number of received PFC frames with PFC enabled bit for Pri 5 on a port.	port	All Devices
rx_pfc_ena_frames_pri6	Number of received PFC frames with PFC enabled bit for Pri 6 on a port.	port	All Devices
rx_pfc_ena_frames_pri7	Number of received PFC frames with PFC enabled bit for Pri 7 on a port.	port	All Devices
tx_pfc_frames	Number of PFC frames transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri0	Number of PFC frames with PFC enabled bit for Pri 0 transmitted on a port.	port	All Devices

Extended Port Stats	Description	Type	Available
tx_pfc_ena_frames_pri1	Number of PFC frames with PFC enabled bit for Pri 1 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri2	Number of PFC frames with PFC enabled bit for Pri 2 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri3	Number of PFC frames with PFC enabled bit for Pri 3 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri4	Number of PFC frames with PFC enabled bit for Pri 4 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri5	Number of PFC frames with PFC enabled bit for Pri 5 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri6	Number of PFC frames with PFC enabled bit for Pri 6 transmitted on a port.	port	All Devices
tx_pfc_ena_frames_pri7	Number of PFC frames with PFC enabled bit for Pri 7 transmitted on a port.	port	All Devices

Port Drop Statistics Counters

The port drop statistics counters are shown in the following table.

Table 51: Port Drop Statistics Counters

Extended Port Stats	Description	Type	Available
tx_stat_discard	Number of transmit packets discarded at the network port due to insufficient network side buffer resources.	port	All Devices
tx_stat_error	Number of packets dropped at the network port due to errors (including the errors at the MAC/PHY level).	port	All Devices
rx_stat_discard	Number of packets discarded at the network port due to lack of network side buffer resources.	port	All Devices
rx_stat_err	Number of packets dropped at the network port due to errors (including the errors at the MAC/PHY level).	port	All Devices

Acronyms

The acronyms are shown in the following table.

Table 52: Acronyms

Acronym	Description
FCS	Frame Check Sequence
MTU	Maximum Transmission Unit
HOL	Head Of Line
VLAN	Virtual Local Area Network
PFC	Priority-Based Flow Control
EEE	Energy Efficient Ethernet
LPI	Low Power Idle
PRI	Priority (field of VLAN)
RoCE	RDMA over Converged Ethernet
CoS	Class of Service
TPA	TCP Payload Aggregation
DLLP	PCIe Data Link Layer Packet
TLP	PCIe Transaction Layer Protocol
CRC	Cyclical Redundancy Check
LCRC	PCIe Link CRC
LTSSM	Link Training and Status State Machine

Ethernet Network Adapter Utilities

Provides information on various Ethernet network adapter utilities.

This section contains information on all of the available utilities for Broadcom Ethernet network adapters.

- NICCLI
- Linux Installer
- NIC Apps
- NIC Tune
- RoCE Analyzer
- SOS Reporting Tool

NICCLI

Provides information on the NICCLI configuration utility.

The NICCLI configuration utility sets the nonvolatile configuration elements of the Broadcom Ethernet network adapter, such as enabling or disabling RoCE, SR-IOV, and other options. The NICCLI configuration utility can also perform firmware upgrades. The NICCLI configuration utility uses the L2 driver in Linux, VMWare. FreeBSD, and Windows. In the UEFI environment, NICCLI uses the PCI bar to communicate with the hardware. The NICCLI configuration utility supports both the BCM9574XX, BCM95750X, and BCM957608 family of devices.

Note: To use the NICCLI configuration utility, log in as `root` or use `sudo` as part of the command in Linux and run as `Administrator` in Windows.

This section is divided into the following sections:

- Installing the NICCLI Configuration Utility
- Using the NICCLI Configuration Utility
- NICCLI Logging
- NICCLI Command Line Options
- NICCLI Configuration Utility Commands
- VMWare Signed and Unsigned Command Mapping
- Mapping NICCLI Commands (233 to 234)

Installing the NICCLI Configuration Utility

Provides installation instructions for the NICCLI configuration utility.

Supported Platforms for NICCLI

Table 53: Supported Platforms for NICCLI

Operating System	Distribution
Linux	x86_64, aarch64
Windows	x86_64
ESXi 8/9	x86_64
FreeBSD	x86_64

Operating System	Distribution
UEFI	x86_64, aarch64

Installing the NICCLI Utility in Linux

This section provides information on installing and executing the NICCLI configuration utility in Linux.

Installing the NICCLI Package in Linux

There are two ways to install the NICCLI utility in Linux:

- NICCLI RPM
  ```
  sudo rpm -i niccli -<version>.rpm
  ```
- NICCLI DEB
  ```
  sudo dpkg –i niccli-<version>.deb
  ```

Executing the NICCLI Utility in Linux

Use `niccli-235.xxx` to execute the NICCLI binary using the package without installing the RPM/DEB package. The package must be unzipped before executing `niccli.<arch>` on the operating system.

Installing the NICCLI Utility in Windows

To install the NICCLI configuration utility, unzip the provided Windows package file and use the `niccli.exe` file to run it in Windows.

Installing the NICCLI Configuration Utility in VMware

This section provides information on installing and executing the NICCLI configuration utility in VMware.

Note: For the ESXi 8.x and 9.x, when the signed niccli package is used, all niccli commands must have `esxcli` added to the beginning of each command. For example, esxcli niccli --list_devices.

There are two ways to install the NICCLI utility in VMware:

- VIB Package
  ```
  esxcli software vib install -v <VIB package> --no-sig-check
  ```
  Note: The VMware signing of the vib package is in process.
- Zip Package
  ```
  esxcli software vib install -d <zip package>
  ```

Installing the NICCLI Configuration Utility in FreeBSD

To install the NICCLI configuration utility, unzip the provided FreeBSD package file and use the `niccli.freebsd` file to run it in FreeBSD.

Installing the NICCLI Configuration Utility in UEFI

To install the NICCLI configuration utility, unzip the provided UEFI package file and use the `niccli<arch>.efi` file to run it in UEFI shell.

Using the NICCLI Configuration Utility

Provides instructions for using the NICCLI configuration utility software for Ethernet network adapters for Linux, VMware, Windows, and FreeBSD.

NICCLI Configuration Utility Interface and Usage

The NICCLI Configuration Utility is a management tool that is used to perform operations on Broadcom Ethernet network adapters. This utility provides support for PCI and operational Inband communication. The utility also accepts arguments to select the communication interface or the specified device in which to communicate from the device list.

The NICCLI configuration utilities provide three different types of interfaces. By default, the utility starts with the interactive interface. The utility accepts three groups of command arguments based on the existing CLI standards.

```
<niccli> <HW i/f argument> [util arguments] [Target command]
```

NICCLI Configuration and Usage on VMWare 8.x

The following syntax is used when the signed bundle of NICCLI is installed:

Syntax

```
esxcli niccli <command> -c <connection_type> -v <connection_type_value> [command options]
```

Parameters

-c

Indicates connection type connection_type : Value for connection type. Supported values are [dev|i|pci]

-v

Indicates connection type value connection_type_value : Supported values are index_number, PF MAC Address and PCI Address

Examples

1. `esxcli niccli list`
2. `esxcli niccli debug -c dev -v 1 --coredump`
3. `esxcli niccli link -c dev -v BC:97:E1:70:14:10 --status`
4. `esxcli niccli debug -c pci -v 0000:86:00.00 --coredump`

Hardware Interface Group of Arguments

The interface arguments depend on the hardware connection type and its specified depending arguments. The NICCLI configuration utility supports the `-pci` interface which takes the PCI Bus/Device/Location of the device. Alternatively, the utility also offers to list all the available Ethernet network adapter PCI devices in the system along with the appropriate Ethernet/Network interface names.

The NICCLI configuration utility has the `-i/-dev` index support which can select when more than one device is found within the host.

NICCLI Configuration Utility Arguments

The utility arguments are optional. These are specific to the NICCLI configuration utility itself, for example, convert all the output into JSON or increase the logger verbosity, and so forth.

Target Command

The target is nothing but the NICCLI configuration utility connected device. These targets offer a specific set of commands depending on the connected interface/device. The target-specific commands are executed upon acquiring the connection with the target.

NICCLI Configuration Utility Commands

All the commands that are provided are case-sensitive and operate with any of the interface modes. The following rules are for the newly defined NICCLI configuration utility syntax. The commands use a specific syntax as follows:

- `< >` mandates user to specify the value
- `[ ]` is an optional parameter.
- Parameter syntaxes can also be combined such as `[ –i <index value>]` optional `–i` index argument but mandatory index value, if `–i` switch specified.
- The NICCLI configuration utility provides "help" command with brief information for every command.
- The NICCLI configuration utility accepts combinations such as -h, -?, '--help' to display the help.
- Every command also has a detailed help description.
- The NICCLI configuration utility also displays the supported commands and/or valid command syntax when the user executes an invalid command. The NICCLI configuration supports the user command line argument as follows:

```
niccli –i <index> <command line>
niccli --pci <domain:bus:device.function> <command line>
```

NICCLI Configuration Utility Help

To access the NICCLI configuration utility help, use the following command:

```
niccli -h|--help
```

Example:

```
niccli --help
```

The utility provides three modes of execution as shown in the following sections:

Oneline Mode

Execute the NICCLI configuration utility on a per-target command basis. In this mode, specify the hardware interface and target command with appropriate arguments. The NICCLI configuration utility connects to the target, executes the target command, and exits from the application. The return status of the command is the exit status of the NICCLI configuration utility.

To list the available targets for Oneline mode use the following command:

```
niccli --list
```

Use the following command to display the list of available commands for Oneline Mode:

```
niccli [-i <index of the target> | --pci <NIC pci address>] --help
```

Use the following command to display the help for a specific command:

```
niccli [-i [<index of the target> | <mac addr> | <NIC pci address>]] --help <command>
```

Example:

```
niccli.x86_64 -i 3 --help nvm
```

Interactive Mode

The NICCLI configuration utility starts in interactive console mode if no target command is provided. The interface starts with the target prompt upon a successful connection with the target.

- The NICCLI configuration utility provides a `--help` command to list all the available or supported commands.
- The NICCLI configuration utility supports `--help|-h [command]` to display detailed help for the specific command. Upon execution of the given user command, the prompt is shown again for the next command.

Note: This mode is best suited for connecting to the target and executing multiple operations/commands without having to disconnect from the target. This improves performance and time in establishing a connection with the target each time while executing a command. This is only for interactive usage and is not designed or meant for the scripting.

To launch in Interactive Mode, use the following command:

```
<NIC CLI executable> [-i <index of the target> | --pci <NIC pci address>]
```

Use the following command to display the list of available commands for Interactive Mode:

```
'-h|--help'
```

Batch Mode

Write the list of commands into a flat text file and execute them in the NICCLI configuration utility without disconnecting. This combines Interactive and OneLine modes without disconnecting the target. If any one of the commands fails, the NICCLI configuration utility exits and shall not continue to execute the rest of the commands from the script.

To launch in Batch Mode, use the following command:

```
<NIC CLI executable> [-i <index of the target> | -pci <NIC pci address>] --batch <batch file>
```

Note: Batch mode requires a flat text file with utility-supported commands. Supported commands can be listed using OneLine mode or Interactive mode. Upon failure of any commands, the utility exits without continuing with other commands.

NICCLI Logging

Provides information on NICCLI logging.

The `nicclilog.ini` file is available in the niccli tool package and it captures the niccli logs. Specify any of the following options in the CLI command:

- niccli --verbose NULL <command>– Prints the verbose logs on the console.
- niccli --debug NULL <command> – Prints the debug logs on the console.
- niccli --verbose file.txt <command> – Redirects the verbose logs into the file specified.
- niccli --debug file.txt <command> – Redirects the debug logs into the file specified.

Note: Environmental Support:

1. NICCLI logging is not supported on VMWare operating systems when a signed NICCLI plugin file is installed.
2. Default logging is disabled by assigning the value 0 to `DEBUG_LOG_LEVEL_ALL` in the `nicclilog.ini` file.
3. In the UEFI environment, continuous file write calls are synchronous and blocks execution until all data is transmitted.
4. If the system is still in an early boot or pre-OS state, there is no buffering or optimization. This is a limitation from the preboot environment itself.

Logging Location

By default, the logs are stored in the following locations:

- Linux, FreeBSD – /var/log
- Windows – C:\ProgramData\Broadcom\logs
- EFI – Current location

Disabling NICCLI Logging

To disable NICCLI logging:

1. Add 0 to `DEBUG_LOG_LEVEL_ALL` in the `nicclilog.ini` file.

Sample nicclilog.ini File

```
--------------------------------------------------------------------------
; Configuration file for NICCLI/NICCLID/NICCLIF/NICCLILOM
; Created: 2025-03-21
; Logging Level 0 (OFF) 1(DEBUG), 2(VERBOSE), 4(FUNC)
; Logging level can be combined. For E.g.
; DEBUG_LOG_LEVEL_ALL = 3 (1(DEBUG), 2(VERBOSE))
[Logging Level]
DEBUG_LOG_LEVEL_ALL=3
; Logging type can either NULL, DEFAULT, Log_file
; NULL : Displays the debug prints on the terminal
; or command prompt.
; DEFAULT : Log in default location. (For Eg. /var/log)
; Default file name : <utility_name>_d_<time_stamp>.log
; Log_file : Provided file name will be logged.
[Logging Type]
DEBUG_LOG_FILE=DEFAULT
--------------------------------------------------------------------------
```

NICCLI Command Line Options

Provides information on supported NICCLI command options.

To get a current list of supported commands by the NICCLI utility, use the following command:

```
niccli -h|-help|help
```

Options

-i|--index

Index of the target.

--search

Searches for a keyword(s) to match the relevant commands.

--pci

NIC PCI address.

--dev

Index or MAC or NIC PCI address or NIC interface.

-j|--json

List the output in JSON format.

Table 54: Commands Commands – Interface/Device Not Required

Command	Description
-v|--version	Displays the version of the utility.
-h|--help	Lists the available commands.
-l|--list	Lists all the supported devices with target indexes.
-d|--list_devices	Lists all the supported devices with basic information.
-e|--list_ethernet	Lists all the supported device interface names.

Command	Description
`--batch`	Enters into batch mode.
`devid`	Query Broadcom device ID's.
`quit`	Quits from the application (Applicable in interactive mode only).

Table 55: Command Categories

Command	Description
show	Displays NIC device related Information.
nvm	Query or Configure device NVM.
fw	Firmware manager.
qos	Query or configure device QOS parameters.
linkdiag	Link Diagnostics.
serdes	Plots the SerDes eyes values.
vf	Performs VF operations.
cable	Display the cable information.
link	Link Operations.
timesync	Peer-to-Peer related operations.
counters	Display and clear the PCIe port counters.
tunnel	Performs Custom, GRE Tunnel and RSS (receive side scaling) operations.
msix	Query or configure MSIX vector of VF's for each PF.
mh	Modify and retrieve the PF count for each PCIe endpoint.
resmgmt	Query and Configure resources of the device.
ccparams	Query or configure the congestion control(cc) parameters for RoCE.
debug	Dumps device internal configuration registers.
pcie	Query and configure the PCIe operations.

NICCLI Configuration Utility Commands

Provides information on all available NICCLI Utility Commands.

This section provides information on the following NICCLI Utility Commands:

- fw Command
- nvm Command
- qos Command
- linkdiag Command
- serdes Command
- cable Command
- link Command
- timesync Command

- counters Command
- vf Command
- tunnel Command
- msix Command
- mh Command
- resmgmt Command
- ccparams Command
- debug Command
- show Command
- pcie Command

fw Command

The `fw` command performs firmware operations.

Note: Operating System Support:

- The `online` option is supported only on Linux and Windows operating systems.
- The `recovery` option is only supported on the Linux operating system and UEFI platform.
- The `reset --immediate` option is only supported on Linux operating systems.

Syntax

```
fw <-u|--update> -f <package file> [--force] [-y|--yes]
fw <-u|--update> <--online> [--force] [-y|--yes]
fw <-u|--update> -f <package file> <--recovery> [--force] [-y|--yes]
fw <--reset> [--immediate]
fw <-l|--livepatch> <--show>
fw <-l|--livepatch> <-a|--activate> [target_fw]
fw <-l|--livepatch> <-d|--deactivate> [target_fw]
fw <-l|--livepatch> <-p|--patch_update> [target_fw] -f <patch file>
```

Parameters

`-u|--update`

Perform the firmware install/update.

`-f|--file`

This option is to provide the package file.

`--force`

Forces the installation of the package file.

`--reset`

This option is to perform the reset operations.

`-y|--yes`

Answer as `yes` in prompts.

`--immediate`

This option resets the device without reloading the drivers.

`-l|--livepatch`

Perform the firmware live patch operations.

--show

Show the livepatch target firmware versions.

-a|--activate

Activate the firmware livepatch from the NVM.

-d|--deactivate

Deactivate the firmware livepatch from the NVM.

-p|--patch_update

Update the patch file directly to the device i.e. without installing it in NVM

target_fw

Target firmware is an optional parameter to active/deactivate the livepatch. By default, the tool updates all the supported target firmwares. The `target_fw` strings supported are `common_fw` or `secure_fw` on BCM95750X devices. The `chimp_fw` string is supported on BCM9574X devices.

--online

Retrieve the firmware image online from the Broadcom web server and perform the update.

-r|--recovery

Recovers the adapter and updates the package file.

Examples

To perform the firmware update:

```
niccli -i 1 fw --update -f FW.pkg --yes
```

To perform the online firmware update:

```
niccli -i 1 fw --update --online --yes
```

To perform the basic adapter recovery:

```
niccli -i 1 fw --update -f FW.pkg --recovery
```

To recover the adapter from the bricked or non-operational state:

```
niccli -i 1 fw --update -f FW.pkg --recovery --force
```

To reset the device:

```
niccli -i 1 fw --reset
```

To reset the device without loading the drivers:

```
niccli -i 1 fw --reset --immediate
```

To query the livepatch firmware versions:

```
niccli -i 1 fw --livepatch --show
```

To activate the livepatch firmware from NVM:

```
niccli -i 1 fw --livepatch --activate
```

To deactivate the livepatch firmware from NVM:

```
niccli -i 1 fw --livepatch --deactivate
```

To update the patch file directly to the device:

```
niccli -i 1 fw --livepatch --patch_update -f patch.pkg
```

nvm Command

The `nvm` command provides the following configuration options for the device:

- Displays the current settings of the NVM configuration.
- Configures the current settings of the NVM configuration.
- Saves the current settings of the NVM configuration.

Note: Operating System Support:

- The `view` option is supported on Linux, Windows, VMWare, and FreeBSD operating systems.
- The `backup [--cfg]` option is supported on Linux, Windows, VMWare OS, and UEFI Platform.

Syntax

```
nvm --view [-V] [-f <firmware package file name>] [-t|--type <nvm directory name>]
nvm -l|--list [-V] [-f <firmware package file name>]
nvm --verify [-V] [-f <firmware package file name>]
nvm -n|--sync
nvm -F|--restore_factory_defaults [--silent]
nvm -r|--dir_read -f <file name> -t|--type <nvm directory name>
nvm -w|--dir_write -f <file name> -t|--type <nvm directory name>
nvm -S|--saveoptions -f <file name>
nvm -O|--optionhelp <option name>
nvm -g|--getoption <option name> [--scope <scope index>]
nvm -s|--setoption <option names with comma separated>
-v|--value <option value with comma separated> [--scope <scope index>]
nvm -b|--backup [--cfg]
nvm -L|--listoptions --diff
```

Parameters

`--view`

View the NVM item data

`-l|--list`

Display the NVM components and its associated version details.

`--verify`

Verify packages and NVM.

`-F|--restore_factory_defaults`

Restores NVM configuration to factory defaults.

`-n|--sync`

Synchronize SBI, SRT and CRT Primary and Secondary FW images. Supported on BCM9575xxx and BCM9576xxx devices.

`-r|--dir_read`

Read the NVM item data and write its contents to a file.

`-w|--dir_write`

Create or overwrite NVM data item with a file.

-S|--saveoptions

Save NVM configuration options on the device to a file. Only the end user access NVM configuration options are saved.

-O|--optionhelp

Detailed help for the NVM configuration option.

-g|--getoption

Get NVM configuration option of a device.

---setoption

Set NVM configuration option of a device.

-v|--value

The value for the specified option. Value can be in hex or decimal format.

--scope

The scope can be either of function or port index.

-L|--listoptions

Displays current and default NVM configuration options of a device.

--diff

Displays the difference between current and default NVM configuration options of a device.

--silent

Silent option. Does not prompt for user message.

-V

Verbosity.

-f

Input file name.

-t|--type

Input NVM directory name string.

Examples

To view the NVM directory entries in detail:

```
niccli -i 1 nvm --view
```

To list the NVM directory entries:

```
niccli -i 1 nvm -l -f BCM957608-P2200GQF00.pkg
```

To verify the NVM directory entries:

```
niccli -i 1 nvm --verify
```

To sync the primary and secondary firmware images:

```
niccli -i 1 nvm -n
```

To read NVM directory:

```
niccli -i 1 nvm -r -t pkglog -f data.txt
```

To write NVM directory:

```
niccli -i 1 nvm -w -f data.txt -t pkglog
```

To restore the config to factory defaults:

```
niccli -i 1 nvm --restore_factory_defaults --silent
```

To save the NVM config to a file:

```
niccli -i 1 nvm --saveoptions -f output.txt
```

To display the current settings for the NVM option:

```
niccli -i 1 nvm --getoption mac_address --scope 0
```

To configure the current settings of the NVM configuration:

```
niccli -i 1 nvm --setoption an_protocol --value 1 --scope 0
niccli -i 1 nvm -s 1,1 --scope 0,1 -v B0:26:28:99:88:14,B0:26:28:99:88:15
```

To list the current and default config settings:

```
niccli -i 1 nvm -L --diff
```

To get the help of each NVM config:

```
niccli -i 1 nvm --optionhelp an_protocol
```

To backup the NVM contents to a file:

```
niccli -i 1 nvm --backup
niccli -i 1 nvm --backup --cfg
```

qos Command

The `qos` command is used to query and configure the various QOS parameters such as ETS, priority to traffic class, application TLV's, receive rate control, TX, and RX rate limits transmit (egress) and receive (ingress) buffer threshold input parameters.

Note: Operating System Support:

- This command is supported on Linux, Windows, VMWare, and FreeBSD operating systems.
- The following options `ets` , `pfc` , `up2tc` , `apptlv` , `dscp2prio` , `listma` , `tc` , `rx_port_rate_limit` , `rx_rate_limit` , `rx_ep_rate_limit` , `tx_partition_rate_limit` , `tx_port_rate_limit` , and `tx_ep_rate_limit` are supported only in Linux and FreeBSD operating systems.

Syntax

```
qos <-E|--ets> --show
qos <-E|--ets --tsa <tc[0-7]:[ets|strict], ...> --up2tc <priority[0-7]:tc>, ...> --tcbw <list>
qos --pfc --enable <pfc list>
qos --up2tc --pri <priority[0-7]:tc, ...>
qos --apptlv <-a|--add> [<-d|--del>] --app <priority,selector,protocol>
qos <-D|--dscp2prio>
qos <-l|--listmap> --pri2cos
qos --tc --set <-T|--rate_limit> <list of rate limit>
qos <-r|--rx_port_rate_limit> --set --max <value> [-p|---persistent]
qos <-R|--rx_rate_limit> --show [-p|--persistent]
qos <-X|--rx_ep_rate_limit> --set --ep0 <value> [--ep1 <value>] [--ep2 <value>] [--ep3
<value>] [-p|---persistent]
qos <-t|--tx_partition_rate_limit> --show [-p|--persistent]
```

```
qos <-t|--tx_partition_rate_limit> --set --max <value> [-p|--persistent]
qos <-P|--tx_port_rate_limit> --show
qos <-P|--tx_port_rate_limit> --set -max <value>
qos <-x|--tx_ep_rate_limit> --show
qos <-x|--tx_ep_rate_limit> --set <--port> <port number> --ep0 <value> [--ep1 <value>] [--ep2
<value> [--ep3 <value>] [-p|--persistent]
qos <-n|--ingress> --cosq --show [-p|--persistent]
qos <-n|--ingress> --cosq --set --state <value> [--mode <value>] [-p|--persistent]
qos <-e|--egress> --cosq --show [-p|--persistent]
qos <-e|--egress> --cosq --set --state <value> [-p|--persistent]
qos -g|--timed_tx_pacing_rate_profile --show
```

Parameters

-E|--ets

Queries or Configures enhanced transmission selection, priority to traffic class, traffic class bandwidths, and the list of configured application tlvs.

--tsa

Transmission selection algorithm, sets a comma separated list of traffic classes to the corresponding selection algorithm. Valid algorithms include ets and strict.

--up2tc

Comma separated list mapping user priorities to traffic classes.

--tcbw

Comma separated list of bandwidths for each traffic class the first value being assigned to traffic class 0 and the second to traffic class 1 and so on.

--pfc

Enables priority based flow control on a given priority.

-apptlv

Configures the priority of the application TLV.

-a|--add

Adds the priority of the application TLV.

-d|--del

Deletes the priority of the application TLV.

--app

Key to provide the priority, selector, protocol for configuring the application TLV.

-D|--dscp2prio

Queries the dscp to priority mapping.

-l|--listmap

Lists the priority mapping and related queue id for a given physical function.

--pri2cos

Lists the priority to traffic class mapping.

--tc

Command to set the rate limit for each traffic class.

-T|--rate_limit

Option to provide the comma separated percentage limit for each TC.

-r|--rx_port_rate_limit

Configures the receive side port rate limit

--max

The max option specifies an 8-bit rate limit as a percentage of total link bandwidth with a range of 0 to 100 percent. A value of 0 indicates no rate limit and deletes the previously configured rate limit.

-p|--persistent

Option to write the configuration to NVRAM, but it does not take effect immediately.

-R|--rx_rate_limit

Queries the configured receive side rate control parameters.

-X|--rx_ep_rate_limit

Configures the receive side rate control parameters for a given endpoint.

-t|--tx_partition_rate_limit

Queries and configures the transmit side partition. Rate limit applies to traffic sent from a partition, which is one PF and all of its child VFs.

-P|--tx_port_rate_limit

Queries and configures the transmit side port rate limit.

--port

Specifies the index of the external port of the device.

--ep(x)

Specifies the Tx or Rx endpoints rate limit values.

-x|--tx_ep_rate_limit

Queries and configures the PCIe endpoint transmit rate control.

-n|--ingress

Queries and configures the QoS dynamically to receive buffer thresholds by configuring different input parameters.

-e|--egress

Queries and configures the QoS dynamically at transmit buffer thresholds by configuring different input parameters.

--cosq

This option is used to query or set the cosq parameter, for example, cosq state and the mode.

--state

Bitmask field indicating which traffic classes are enabled or disabled. Each bit represents a specific traffic class, where bit 0 represents traffic class 0 and so on. A value of 0 indicates that the traffic class is not enabled.

--mode

Bitmask field indicating which traffic class is lossy or lossless. Each bit represents a specific traffic class, where bit 0 represents traffic class 0 and so on. A value of 0 indicates that the traffic class is lossy and value 1 indicates that the traffic class is lossless.

-g|--timed_tx_pacing_rate_profile

Query the timed tx pacing rate profile.

Examples

To show the ETS (enhanced transmission selection) configuration:

```
niccli -i 1 qos --ets --show
```

To configure the ETS (enhanced transmission selection):

```
niccli -i 1 qos --ets --tsa 0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict --up2tc
 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0 --tcbw 70,30
```

To enable priority based flow control on a given priority:

```
niccli -i 1 qos --pfc --enable 5,6
niccli -i 1 qos --pfc --enable 0xFF
```

To set the user priorities to traffic classes:

```
niccli -i 1 qos --up2tc 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0
```

To add the priority of the application TLV:

```
niccli -i 1 qos --apptlv --add --app 5,1,35093
```

To delete the priority of the application TLV:

```
niccli -i 1 qos --apptlv --del --app 5,1,35093
```

To query the dscp to priority mapping:

```
niccli -i 1 qos --dscp2prio
```

To list the priority to traffic class mapping:

```
niccli -i 1 qos --listmap --pri2cos
```

To set the rate limit for each traffic class in units of percentage:

```
niccli -i 1 qos --tc --set --rate_limit 10,20,30
niccli -i 1 qos --tc --set --rate_limit 10
```

To configure receive rate control that applies to all traffic in a receive CoS queue group:

```
niccli -i 1 qos --rx_port_rate_limit --set --max 40
niccli -i 1 qos --rx_port_rate_limit --set --max 70 -persistent
```

To show the receive side rate limits:

```
niccli -i 1 qos --rx_rate_limit --show
niccli -i 1 qos --rx_rate_limit --show --persistent
```

To configure endpoint rate limit for all endpoints from one host:

```
niccli -i 1 qos --rx_ep_rate_limit --set --ep0 0
niccli -i 1 qos --rx_ep_rate_limit --set --ep0 0 --persistent
```

To show the TX partition rate limit:

```
niccli -i 1 qos --tx_partition_rate_limit --show
```

To configure the TX partition rate limit:

```
niccli -i 1 qos --tx_partition_rate_limit --set --max 2
```

To show the transmit side port rate limit:

```
niccli -i 1 qos --tx_port_rate_limit --show
```

To show the transmit side port rate limit:

```
niccli -i 1 qos --tx_port_rate_limit --show
```

To configure the transmit side port rate limit:

```
niccli -i 1 qos --tx_port_rate_limit --set --max 2
```

To query the PCIe endpoint transmit rate control:

```
niccli -i 1 qos --tx_ep_rate_limit --port 0 --show
```

To configure the PCIe endpoint transmit rate control for two endpoints:

```
niccli -i 1 qos --tx_ep_rate_limit --set --port 0 --ep0 50 --ep2 40
```

To query the ingress cosq parameters:

```
niccli -i 1 qos --ingress --cosq --show
```

To enable all the 8 traffic classes and mode lossless(1) is configured on traffic class 4:

```
niccli -i 1 qos --ingress --cosq --set --state 255 --mode 16
```

To query the egress cosq parameters:

```
niccli -i 1 qos --egress --cosq --show
```

To configure the egress cosq parameters. Below example enables all the 8 queues:

```
niccli -i 1 qos --egress --cosq --set --state 255
```

To query the timed TX pacing rate profile:

```
niccli -i 1 qos --timed_tx_pacing_rate_profile --show
```

linkdiag Command

The `linkdiag` command is used to perform link diagnostic tests such as PRBS, loopback, DSCDump, and TXFIR settings.

Note: Operating System Support:

- The`fdrstat` options are supported on Linux, Windows, VMWare, and FreeBSD Platforms.
- The`DSC Dump` options are supported on Linux, VMWare, and FreeBSD operating systems.
- The`loopback` options are supported only on Linux operating systems.
- The`PRBS` test options are supported only on the Linux operating system and UEFI Platform.
- The`TxFir` options are supported on Linux, Windows, VMWare, and FreeBSD Platforms

Syntax

```
linkdiag -T|--txfir --show <-M|--modulation_type> <mod_type> <-l|--lane> <lane_number>
linkdiag -T|--txfir --set <-M|--modulation_type> <mod_type> <-l|--lane> <lane_mask>
--pre1 <value> --pre2 <value> [--pre3 <value>] --main <value> --post1 <value>
--post2 <value> [--post3 <value> --amp <value> --nlcl <value> --nlcu <value>]
linkdiag -F|--fdrstat [--start] [--stop] [--clear] [--counters]
linkdiag -D|--dscdump -l|--lane <lane_number> [-a|--diag_level <level>]
linkdiag -L|--loopback --show
linkdiag -L|--loopback [<-P|--phy_remote> | <-p|--phy_local> | <-m|--mac_local> |
```

```
<-d|--disable>]
linkdiag -L|--loopback [--external] [--RJ45]
linkdiag -P|--prbs_test <-e|--enable | -d|--disable> [--mode <mode_value>]
[<-r|--rx_lane_mask> <value>]
[<-t|--tx_lane_mask> <value>] [<-s|--duration> <value in seconds>] [--tcode]]
```

Parameters

-T|--txfir

This option is used to query and configure the TX FIR (transmitter finite impulse response).

-F|--fdrstat

This option is used for FDR (Flight Data Recorder) to collect the FEC Performance data.

-D|--dscdump

This option is used to retrieve DSC dump data from a device.

-L|--loopback

This option is used to query and configure the different loopback Modes, for example, PHY loopback, mac loopback, and external loopback.

-P|--prbs_test

This option is used in port interface debugging to analyze the quality of the link. The test can be run on a port or per lane with a specific polynomial.

Note: The interface(s) should be fully initialized before the testing. During testing, there should not be any queries or configs sent to the card.`ifdown` the interface(s) is recommended.

-M|--modulation_type

Modulation types of TxFIR. Supported values are 'NRZ','PAM4','C2MNRZ','C2MPAM4', 'PAM4-112','C2MPAM4-112G', and 'LPOPAM4-112G' of the device. The modulation types 'PAM4-112','C2MPAM4-112G', and 'LPOPAM4-112G' are only supported on BCM957608 devices. The modulation types 'PAM4' and 'C2MPAM4' are supported on BCM95750X and BCM957608 devices.

-l|--lane

TXFIR show command takes the MRS lane number and lane mask for the TXFIR configuration. To collect the dsc dump on all the supported lanes, please provide a value of 65535. DSC dump on all lanes is supported only on BCM957608 devices.

--pre1

This is a mandatory parameter to configure the TXFIR settings. This parameter is supported for all the modulation types and the valid range is from -32768 to 32767.

--pre2

This is a mandatory parameter to configure the TXFIR settings. This parameter is supported for all the modulation types and the valid range is from -32768 to 32767.

--main

This is a mandatory parameter to configure the TXFIR settings. This parameter is supported for all the modulation types and the valid range is from -32768 to 32767.

--post1

This is a mandatory parameter to configure the TXFIR settings. This parameter is supported for all the modulation types and the valid range is from -32768 to 32767.

--post2

This is a mandatory parameter to configure the TXFIR settings. This parameter is supported for all the modulation types and the valid range is from -32768 to 32767.

--pre3

This is an optional parameter to configure the TXFIR settings. This parameter is supported on following modulation types, for the example, 'LPOPAM4-112G', 'PAM4-112', 'C2MPAM4', and 'C2MPAM4-112G'. The valid range is from -32768 to 32767.

--post3

This is an optional parameter to configure the TXFIR settings. This parameter is supported on the following modulation types, for example, 'NRZ','PAM4' and 'C2MNRZ'. The valid range is from -32768 to 32767.

--amp

This is an optional parameter to configure the TXFIR settings. This parameter is supported on the following modulation types, for example, 'NRZ','PAM4','C2MNRZ','PAM4-112','C2MPAM4', and 'C2MPAM4-112G'. The valid range is from -32768 to 32767.

--nlcu

This is an optional parameter to configure the TXFIR settings. This parameter is supported only on 'LPOPAM4-112G' modulation type. The valid range is from -100 to 100.

--nlcl

This is an optional parameter to configure the TXFIR settings. This parameter is supported only on 'LPOPAM4-112G' modulation type. The valid value is 0.

--start

This is an optional parameter to start the fdrstat.

--stop

This is an optional parameter to stop the fdrstat.

--clear

This is an optional parameter to clear the fdrstat.

--counters

This is an optional parameter to pull the fdrstat counters information.

-a|--diag_level

This is an optional parameter. If the user does not specify this parameter by default DSC dump will be collected on all the supported diag levels. Supported diag levels are as follows: 0 = diag lane 1 = diag core 2 = diag event 3 = diag eye 4 = diag reg core 5 = diag reg lane 6 = diag uc core 7 = diag uc lane 8 = diag lane debug 9 = diag ber vert 10 = diag ber horz 11 = diag event safe 12 = diag timestamp

-R|--phy_remote

This option enables loopback of local PHY Rx to peer PHY Tx. The packets transmitted by peer are looped back to the peer at the PHY. No packets will reach the host. Host will see a link down.

-p|--phy_local

This option enables a loopback of local TX to local RX at the PHY. Any packets transmitted from the host are looped back to the host. No packets will be transmitted on the line. If any peer is connected, the peer should ignore the link status from the host.

-m|--mac_local

This option enables a loopback of local TX to local RX at the MAC. Any packets transmitted from the host are looped back to the host. No packets will be transmitted on the line. If any peer is connected, the peer should ignore the link status from host.

-d|--disable

This option disables the current loopback settings. In case of prbs_test this option disables the PRBS test.

--external

This option prepares the PHY from external loopback and suppresses NONCE generation for auto negotiation to work with an external loopback. This option should only be used if the same port external loopback dongle or equivalent is used. Without this option and AN enabled, the host may not see a link up.

--RJ45

This test is designed for dual port 10GBase-T where the PRBS test is not applied. A compliant UTP cable is needed to connect between the two ports of the controller because auto-negotiation is required to establish a link, traffic is initiated from one port. The PHY of the other port is put into a remote loopback mode, essentially serving as a loopback plug, to bounce packets back to the initiating port. This test requires the production Linux driver (`bnxt_en`) to be loaded in order to execute. Packets are sent and received through the Linux OS network stack

-e|--enable

Enable the PRBS test.

--mode

Specify the supported modes. And the supported modes are 'PRBS31','PRBS7','PRBS9','PRBS11','PRBS15','PRBS23','PRBS58','PRBS49','PRBS10','PRBS20' and 'PRBS13'. The default mode value is PRBS31

-t|--tx_lane_mask

Transmitter lane mask value.

-s|--duration

Duration to run the prbs test. Default time is 10 seconds.

--tcode

If this option was provided. The prbs test will run on t-code project as well.

Examples

To set the TXFIR settings:

```
niccli -i 1 linkdiag -T --set --modulation_type LPOPAM4-112G --lane 1 --pre1 1 --pre2 -2 --pre3 10 --main 12
 --post1 -10 --post2 15 --nlcl 0 --nlcu 1
niccli -i 1 linkdiag -T --set --modulation_type PAM4 --lane 1 --pre1 1 --pre2 -2 --main 12 --amp 10 --post1
 -10 --post2 15 --post3 10
niccli -i 1 linkdiag -T --set --modulation_type NRZ --lane 1 --pre1 1 --pre2 -2 --main 12 --amp 10 --post1 -10
 --post2 15 --post3 10
niccli -i 1 linkdiag -T --set --modulation_type PAM4-112 --lane 1 --pre1 1 --pre2 -2 --pre3 5 --main 12 --amp
 10  --post1 -10 --post2 15
```

To get the TXFIR settings:

```
niccli -i 1 linkdiag -T --show --modulation_type LPOPAM4-112G --lane 0
niccli -i 1 linkdiag -T --show --modulation_type PAM4 --lane 0
niccli -i 1linkdiag -T --show --modulation_type NRZ --lane 0
niccli -i 1linkdiag -T --show --modulation_type PAM4-112 --lane 0
```

To Start the fdrstat:

```
niccli -i 1 linkdiag --fdrstat --start
```

To Stop the fdrstat:

```
niccli -i 1 linkdiag --fdrstat --stop
```

To Clear the fdrstat:

```
niccli -i 1 linkdiag --fdrstat --clear
```

To Pull the fdrstat counters:

```
niccli -i 1 linkdiag --fdrstat --counters
```

To get the DSC Dump:

```
niccli -i 1 linkdiag -D --lane 0
niccli -i 1 linkdiag -D --lane 0 --diag_level 2
```

To get loopback status:

```
niccli -i 1 linkdiag --loopback --show
```

To disable the loopback mode:

```
niccli -i 1 linkdiag --loopback --disable
```

To enable the `mac_local` loopback mode:

```
niccli -i 1 linkdiag --loopback --mac_local
```

To enable the phy_local loopback mode:

```
niccli -i 1 linkdiag --loopback --phy_local
```

To enable the phy_remote loopback mode:

```
niccli -i 1 linkdiag --loopback --phy_remote
```

To enable the external loopback mode:

```
niccli -i 1 linkdiag --loopback --external
```

To enable the external RJ45 loopback mode:

```
niccli -i 1 linkdiag --loopback --external --RJ45
```

To enable the PRBS Test with default:

```
niccli -i 1 linkdiag --prbs_test --enable
```

To disable the PRBS Test:

```
niccli -i 1 linkdiag --prbs_test --disable
```

To run the PRBS Test with user provided params:

```
niccli -i 1 linkdiag --prbs_test --enable --mode PRBS31 --rx_lane_mask 255 --tx_lane_mask 255 --duration 10
```

serdes Command

The `serdes` command is used to plot the SerDes ethernet eye, PCI eye scope, and margin values of the eye.

Note: While plotting the SerDes ethernet eye, link toggling is expected. Both the SerDes ethernet and PCI eye share the resources of the NIC. Therefore, these commands cannot be run concurrently. If PCI eye is running and the ethernet eye tool is run, it will return a failure.

Note: Operating System Support:

- This command is supported only on Linux operating systems.

Syntax

```
serdes --eye -e|--ethernet [-l|--lane <ethernet lane number>] [<-P|--plot>]
serdes --eye -p|--pci <-l|--lane> <pci_lane_number> [-P|--plot] [-t|--target_ber <value>]
serdes --eye -p|--pci -s|--stop
```

Parameters

--eye

This option is used to plot the SerDes PCI and ethernet eye.

-e|--ethernet

This option is used to plot the SerDes ethernet eye. By default this option displays only horizontal and vertical margin values, including the test result.

-p|--pci

This option is used to plot the SerDes PCI eye.

-l--lane

This option is used to specify the lane number. For PCIe SerDes test, the maximum value is the device PCIe lane width minus 1. Valid values are from 0 to 15. For ethernet serdes test the valid range is from 0 to 7.

-P|--plot

This is an optional parameter. When user specifies this option, will plot the eye and displays the horizontal and vertical margin values, including the test result.

-t|--targetber

This option is used to specify the target bit error rate. By default SerDes pCI eye is plotted with BER "1e-8". This option is only supported on BCM95750X and above devices. The supported target BER values are "1e-8", "1e-9", "1e-10" and "1e-11".

-s|--stop

This option is used to stop the running SerDes PCI eye plotting. This option is only supported on BCM95750X and above devices.

Examples

To plot the ethernet SerDes eye:

```
niccli -i 1 serdes --eye -e -P
niccli -i 1 serdes --eye --ethernet --plot
```

To plot the ethernet serdes eye with specific lane number:

```
niccli -i 1 serdes --eye -e -l 0 -P
niccli -i 1 serdes --eye --ethernet --lane 0 --plot
```

To get the PCI SerDes eye margins and Rx settings with specific lane number:

```
niccli -i 1 serdes --eye --pci --lane 0
niccli -i 1 serdes --eye -p -l 0
```

To plot the PCI SerDes eye with specific lane number:

```
niccli -i 1 serdes --eye --pci --lane 0 --plot
niccli -i 1 serdes --eye -p -l 0 -P
```

To plot the PCI serdes eye with specific lane number and target BER:

```
niccli -i 1 serdes --eye --pci --lane 0 --plot --target_ber 1e-8
niccli -i 1 serdes --eye -p -l 0 -P -t 1e-8
```

To stop the PCI serdes eye test:

```
niccli -i 1 serdes --eye --pci --stop
niccli -i 1 serdes --eye -p -s
```

cable Command

The `cable` command is used to query/decode module EEPROM information and optical diagnostics.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
cable -m|--module_info --show
cable -r|--read_module_eeprom [-p|--page_number <page number> -o|--offset <byte offset>
-l|--length <number of bytes> -b|--bank <bank number> -i|--i2c_address <i2c addr>]
cable -w|--write_module_eeprom -p|--page_number <page number> -o|--offset <byte offset>
-v|--value <bytes>
cable -M|--module_loopback -t|--loopback_type <type> [--lane <module_lane_number>]
```

Parameters

-m|--module_info

Retrieves the module information.

-r|--read_module_eeprom

Reads the module EEPROM in hex format

-w|--write_module_eeprom

Writes the bytes into the module EEPROM

-i|--i2c_address

I2C address of a page. Value less than 0x7f expected. Most of the EEPROMs use 0x50 or 0x51

-p|--page_number

The page number that is being accessed over I2C

-l|--length

Length of EEPROM data to read or write.

-o|--offset

Offset within the page that is being accessed over I2C.

-b|--bank

The bank number of the page that is being accessed over I2C.

-v|--value

Bytes to write into module EEPROM.

-M|--module_loopback

This option is used to perform the various module loopback. This operation is supported only on CMIS 4.0 and above supported modules.

-t|--loopback_type

This option is used to run the module loopback on the specified loopback type. The supported module loopback types are as follows:

1. Media Side Output Loopback
2. Media Side Input Loopback
3. Host Side Output Loopback
4. Host Side Input Loopback

--lane

This is an optional parameter. If user specifies this option with a valid lane number, then the module loopback will be run on specified lane.

--show

Displays the information in detail.

Examples

To show the module information:

```
niccli -i 1 cable -m --show
niccli -i 1 cable --module_info --show
```

To Read the module EEPROM:

```
niccli -i 1 cable -r -p 0 -o 0 -l 128 -b 0 -i 0x50
niccli -i 1 cable --read_module_eeprom --page_number 0 --offset 0 --length 128 --bank 0
--i2c_address 0x50
```

To Write to the module EEPROM:

```
niccli -i 1 cable -w -p 0 -o 0 -v 20
niccli -i 1 cable --write_module_eeprom --page_number 0 --offset 0 --value 20
```

To run the Media Side Output Loopback:

```
niccli -i 1 cable -M -t 1 -l 0
niccli -i 1 cable --module_loopback --loopback_type 1 --lane 0
```

To run the Media Side Input Loopback:

```
niccli -i 1 cable -M -t 2 -l 0
```

```
niccli -i 1 cable --module_loopback --loopback_type 2 --lane 0
```

To run the Host Side Output Loopback:

```
niccli -i 1 cable -M -t 3 -l 0
niccli -i 1 cable --module_loopback --loopback_type 3 --lane 0
```

To run the Host Side Input Loopback:

```
niccli -i 1 cable -M -t 4 -l 0
niccli -i 1 cable --module_loopback --loopback_type 4 --lane 0
```

link Command

The `link` command is used to query the link status, BER information, physical counters, and configure the port state.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Prerequisites

- Run the `linkdiag --fdrstat` --start command to query the link BER and physical counters on BCM957608 devices.
- All the counters must be accumulated since board reset.
- RS corrected errors, RS uncorrectable errors, and CRC Error Frames must be cleared by using the `link --counters --clear` command.
- Effective Physical Errors and Raw Physical Errors can be restarted by using the `linkdiag --fdrstat --stop` command on BCM957608 devices.
- RS corrected errors are referred to as corrected errors from FEC.
- RS uncorrectable errors are referred to as uncorrectable errors from FEC.
- CRC Error Frames are referred to as error frames from CRC.
- Effective Physical Errors are referred to as uncorrectable codewords from FDR on BCM957608 devices.
- Effective Physical BER are referred to as BER based on Effective Physical Errors on BCM957608 devices.
- Raw Physical Errors are referred to as symbol errors from FDR on BCM957608 devices.
- Raw Physical BER are referred to as BER based on Raw Physical Errors on BCM957608 devices.
- When a port is connected to BMC, the firmware does not honor the link toggle via the `port_state` option.
- BER information and physical counters are only supported on BCM957608 devices.

Syntax

```
link -s|--status
link -c|--counters --show
link -p|--port_state <port state value>
link -S|--port_speed [--speed <value>] [--lanes <number_of_lanes>] -a|--autoneg <on/off> [--fec <value>] [-
t|--training <on/off>]
```

Parameters

`-s|--status`

Shows the link status and related information.

-c|--counters

Show physical counters and BER Info.

-p|--port_state

Configures the portstate. The valid values are 0(Down), 1(UP), and 2(Toggle).

-S|--port_speed

Configures the port speed for the number of lanes, Link training, Autoneg, and FEC.

--speed

Option to provide the speed value in MB.

--lanes

Option to provide the number of lanes. The valid values are 1, 2, 4, and 8.

-a|--autoneg

Option to turn the autoneg speed on or off. The valid values are on or off.

-t|--training

Option to turn link training on or off. The valid values are on or off.

--fec

Option to provide the FEC type. The valid values are:

1.
2. FEC Autoneg enable
3. FEC Clause74 enable
4. FEC Clause91 enable
5. FEC `RS544_1XN` enable
6. FEC `RS544_IEEE` enable
7. FEC `RS272_1XN` enable
8. FEC `RS272_IEEE` enable

Examples

To get the link information:

```
niccli -i 1 link -s
niccli -i 1 link --status
```

To get the physical counters and BER information:

```
niccli -i 1 link -c --show
niccli -i 1 link --counters --show
```

To configure the port state to down:

```
niccli -i 1 link -p 0
niccli -i 1 link --port_state 0
```

To configure the port state to up:

```
niccli -i 1 link -p 1
niccli -i 1 link --port_state 1
```

To toggle the port state:

```
niccli -i 1 link -p 2
niccli -i 1 link --port_state 2
```

To set the port link speed to autoneg:

```
niccli -i 1 link --port_speed --autoneg on
```

To set the port link speed to 100G on 4 lanes and associated params:

```
niccli -i 1 link --port_speed --autoneg off --speed 100000 --lanes 4 --fec 3 --training on
```

timesync Command

The `timesync` command provides the following functionality:

- To set duty cycle on TSIO outgoing signal.
- To set the DLL source for PHC.
- Set PTP extended parameters operation. All the parameters are optional.
- Configure and display the synchronous ethernet frequency profile, primary and secondary clock state.
- TSIO operations for the requested PF/VF.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
timesync < -d | --dutycycle> <--period> <value> --up <value>

timesync <--dll> <-s | --source> <value> <-q | --frequency> <value>

timesync <--ptp> --show
timesync <--ptp> --set <-p | --primary_pf> <pid> [<-v | --primary_vf> <vfid>]
[<-P | --secondary_pf> <pfid>] [<-V | --secondary_vf> <vfid>]

timesync <--synce> --show
timesync <--synce> --set <-Q | --frequency_profile> <value> [<-c | --primary_clock_state> <value>] [<-C | --
secondary_clock_state> <value>]

timesync <--tsio> <-t | --tsio_function_pin> <idx> <-u | --pin_usage_string> <value>
<--state> <value>
```

Parameters

-d|--dutycycle

To set duty cycle on TSIO outgoing signal.

--period

Value for period will be treated as in nanoseconds.

--up

Up flag is used to set the duty cycle and should be lesser that period value.

--dll

To set the DLL source for PHC.

-s | --source

The valid values range is 0 to 4.

-q | --frequency

The valid values range is 0 to 3.

-p| --ptp

PTP extended parameters operation.

--show

Displays timesync operation.

--set

Configures timesync operation.

-p | --primary_pf

Primary physical function ID.

-v | --primary_vf

Primary virtual function ID belongs to primary PF ID.

-P | --secondary_pf

Secondary physical function ID.

-V | --secondary_vf

Secondary virtual function ID belongs to secondary PF ID.

--synce

Configure and display the synchronous ethernet frequency profile. primary and secondary clock state. This command is supported only on BCM95750X devices.

-Q | --frequency_profile

Frequency profile for SyncE recovered clock. Supported profiles are "25MHz"

-c | --primary_clock_state

Enable or disable primary clock for PF or port, overriding previous primary clock setting.

-C | --secondary_clock_state

Enable or disable secondary clock for PF or port, overriding previous secondary clock setting.

--tsio

Displays or Configures tsio function capability on the pin.

-t | --tsio_function_pin

Pin Index.Valid Index 0 to 3.

-u | --pin_usage_string

Pin usage string.

--state

Enable/Disable function capability on the pin.

Examples

Setting duty cycle on TSIO outgoing signal:

```
niccli -i 1 timesync -d --period 1 --up 0
niccli -i 1 timesync --dutycycle --period 1 --up 0
```

To configure the DLL source for PHC:

```
niccli -i 1 timesync --dll -s 1 -q 3
niccli -i 1 timesync --dll --source 1 --frequency 3
```

To perform PTP extended parameters operation:

```
niccli -i 1 timesync --synce --show
niccli -i 1 timesync --synce --set -Q 25MHz
niccli -i 1 timesync --synce --set --frequency_profile 25MHz
```

To perform TSIO function capability on the pin:

```
niccli -i 1 timesync --tsio -t 3 -u 1 --state 1
niccli -i 1 timesync --tsio --tsio_function_pin 3 --pin_usage_string 1 --state 1
```

To perform PTP extended parameters operation:

```
niccli -i 1 timesync --ptp --show
niccli -i 1 timesync --ptp --set -p 1 -v 1
niccli -i 1 timesync --ptp --set --primary_pf 1 --secondary_pf 1
```

counters Command

The `counters` command is used to display and clear the PCIe port counters.

Syntax

```
counters -p | --pcie [-V]
counters -c | --clear
```

Parameters

`-p | --pcie`
Displays PCIe Counters.

`-c | --clear`
Clears the port counters

`-V`
Displays the detailed information of the PCIe Counters.

Examples

To display the PCIe port counters:

```
niccli -i 1 counters -p
niccli -i 1 counters --pcie
niccli -i 1 counters --pcie -V
```

To clear the port counters:

```
niccli -i 1 counters -c
niccli -i 2 counters --clear
```

vf Command

The `vf` command is used to performs VF operations.

Note: Operating System Support:

- This command is supported only on Linux operating systems.

Syntax

```
vf <-t|--trust> --set <-v|--vf_index> <idx> <--state> <enable/disable>
vf <-t|--trust> --show <-v|--vf_index> <idx>
vf <-a|--add_ntuple_filter> <-m|--macaddress> <value> <-p|--dest_port> <value>
<-P|--dst_port_mask> <value> <-v|--vf_index> <idx> <-T|--ip_type> <value>
vf <-d|--free_ntuple_filter> <-l|--filter_id> <value>
vf <-M|--peer_mem_map> <--hpa> <list of values> <--gpa> <list of values>
<--size> <list of values>
```

Parameters

-t|--trust

Perform the trusted VF operations.

-v|--vf_index

Provide the VF index.

--state

Option to enable or disable the trusted VF.

-a|--add_ntuple_filter

Option to add the ntuple flow filter.

-d|--free_ntuple_filter

Option to free the ntuple flow filter.

-m|--macaddress

MAC address in format xx:xx:xx:xx:xx:xx.

-p|--dest_port

Option to provide the destination port.

-P|--dst_port_mask

Option to provide the destination port mask.

-T|--ip_type

Option to provide the IP type. The valid values are:

1. IPV4
2. IPV6
3. ARP-REPLY

-M|--peer_mem_map

Option to configure the GPU host and guest physical address mapping.

--hpa

This option is used to specify the list of host physical addresses. Max 8 entries are supported. User has to provide the list with a comma separated for each host physical address. The value should be in the hex-decimal.

--gpa

This option is used to specify the list of guest physical addresses. Max 8 entries are supported. User has to provide the list with a comma separated for each guest physical address. The value should be in the hex-decimal.

--size

This option is a comma separated list in kilobytes for each mapping. Max 8 entries are supported. The value should be in the hex-decimal.

Examples

To query trusted vf state:

```
niccli -i 1 vf --trust --show --vf_index 0
```

To enable trusted vf state:

```
niccli -i 1 vf --trust --set --vf_index 1 --state enable
```

To add ntuple flow filter for the specified MAC and destination port:

```
niccli -i 1 vf --add_ntuple_filter --macaddress 00:01:02:03:04:a3 -p 1023 -P 0xFFFF -v 1 -T 1
```

To free ntuple flow filter for the specified filter ID:

```
niccli -i 1 vf --free_ntuple_filter --filter_id F06C0000D6C22414
```

To configure the GPU host and guest physical address mapping:

```
niccli -i 1 vf --peer_mem_map --hpa
```

tunnel Command

The `tunnel` command is used to perform custom, GRE Tunnel, and RSS (receive side scaling) operations.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
tunnel --cfg --vxlan <-t|--type> <ipv4|ipv6> --show
tunnel --cfg --vxlan <--add> <-t|--type> <ipv4|ipv6> <-p|--dest_port> <value>
tunnel --cfg --vxlan <--del> <-t|--type> <ipv4|ipv6> <-p|--dest_port> <value>
tunnel --cfg <--rss> --show
tunnel --cfg <--rss> --set <--mode> <inner/outer>
tunnel --cfg <-g|--gre_tunnel_offload> --show
tunnel --cfg <-g|--gre_tunnel_offload> --set --state <enable/disable>
```

Parameters

--cfg

Perform custom, GRE Tunnel and RSS (receive side scaling) operations.

--vxlan

Option to query or configure vxlan type.

-t|--type

Option to provide the IP type. The valid values are "ipv4" and "ipv6".

-p|--dest_port

Option to provide the destination port. Valid range is 0-65535.

--rss

Option to query and configure RSS (receive side scaling).

--mode

Option to configure the RSS mode. The valid values are "inner" and "outer".

-g|--gre_tunnel_offload

Option to query and configure the custom GRE tunnel offload.

--state

Option to enable or disable the non udp port based GRE tunnel offload.

Examples

To show the custom tunnel configuration:

```
niccli -i 1 tunnel --cfg --vxlan --type ipv4 --show
```

To add the custom tunnel configuration on destination port:

```
niccli -i 1 tunnel --cfg --vxlan --add --type ipv4 --dest_port 1024
```

To delete the custom tunnel configuration on destination port:

```
niccli -i 1 tunnel --cfg --vxlan --del --type ipv4 --dest_port 1024
```

To show the RSS mode configuration:

```
niccli -i 1 tunnel --cfg --rss --show
```

To configure the RSS inner mode:

```
niccli -i 1 tunnel --cfg --rss --set --mode inner
```

To show the state of GRE tunnel offload:

```
niccli -i 1 tunnel --cfg --gre_tunnel_offload --show
```

To enable the non UDP port based GRE tunnel offload:

```
niccli -i 1 tunnel --cfg --gre_tunnel_offload --set --state enable
```

msix Command

The `msix` commands are used to query and configure the number of MSI-X max vector values for VF's per each PF.

Note: Operating System Support:

- This command is supported on Linux and Windows operating systems.

Syntax

```
msix -m|--max_vectors --show [--all | --pf <pf number>]
msix -m|--max_vectors --set [--pf <pf number>]
```

Parameters

-m|--max_vectors

Retrieves the register dump crashdump from the firmware.

--pf

PF Number to query the table of 8 rows for msix max vectors.

--show

Get the msix max vectors for PF.

--set

Configures the msix max vectors for PF.

--all

Displays msix max vectors for all the PF's.

Examples

To display the MSI-X max vectors values:

```
niccli -i 1 msix --max_vectors --show --pf 0
```

To display all the values of MSI-X max vectors:

```
niccli -i 1 msix --max_vectors --show --all
```

To configure the MSI-X max vectors values for PF 0:

```
niccli -i 1 msix --max_vectors --set --pf 0
```

mh Command

The `mh` command is used to query and configure the Broadcom Multi-Host PF information.

Note: Operating System Support:

- This command is supported on Linux, Windows, and FreeBSD operating systems.

Syntax

```
mh -p|--pf_alloc --show
mh -p|--pf_alloc --set --ep0 <pf_cnt> --ep1 <pf_cnt> --ep2 <pf_cnt> --ep3 <pf_cnt>
```

Parameters

-p|--pf_alloc

PF Number to query Multi-Host PF information.

--show

Query Multi-Host PF information.

--set

Configure the Multi-Host PF information.

--ep0

Number of PF to be written on EP0.

--ep1

Number of PF to be written on EP1.

--ep2

Number of PF to be written on EP2.

--ep3

Number of PF to be written on EP3.

Note: The number of non-zero values can be for 2 endpoints or 4 endpoints. The sum of all the endpoints should be less than or equal to 16.

Examples

To display the Multi-Host PF information:

```
niccli -i 1 mh --pf_alloc --show
```

To configure the Multi-Host PF information:

```
niccli -i 1 mh --pf_alloc --set --ep0 0 --ep1 0 --ep2 0 --ep3 0
```

resmgmt Command

The `resmgmt` command is used to query and configure resources of the device.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
resmgmt [--pf/--all] [<-p|--profile> | < --min > |< --max > | <-r | --roce_max> | < -m | --max_comple-
tion_rings> | < -s | --strategy >] --show

resmgmt [--pf/--all] --set [<-p|--profile> | < --min > |< --max > | <-r | --roce_max> | < -m | --max_comple-
tion_rings>] [< --bw <bandwidth of each pf with comma separated>] [< -s | --strategy > < minimal/maximal/mini-
mal-static>]
```

Parameters

-p|--profile

Active profile of the device

-r|--roce_max

RoCE enabled PF's on the device (supported on BCM957454 and BCM95741X devices only)

-m|--max_completion_rings

Maximum completion rings for each active PF

s|--strategy

Strategy associated with each active PF.

--min

Minimum bandwidth associated with each active PF.

--max

Maximum bandwidth associated with each active PF.

--show

Displays resources of the device.

--set

Configures resources of the device

--bw

Bandwidth to configure for PF N. This is applicable for min, max, `roce_max` , and `max_cmpl` .

Examples

To query resources of the device:

```
niccli -i 1 resmgmt --all --profile --show
```

To configure resources of the device:

```
niccli -i 1 resmgmt --pf --set --bw 1,2,3 --strategy minimal
```

ccparams Command

The `ccparams` command is used to query and configure the congestion control(cc) parameters for RoCE.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
ccparams -d|--dump
ccparams --set -f <configuration file>
```

Parameters

-d|--dump

Dumps the congestion control parameters into a configuration <xxx>.CFG file. The file is generated in the same directory where the executable is running.

--set

To configure the congestion control parameters using a <xxx>.CFG file.

-f

Configuration file to configure the congestion control (cc) parameters.

Examples

To dump the congestion control params:

```
niccli -i 1 ccparams -d
niccli -i 1 ccparams --dump
```

To configure the congestion control params:

```
niccli -i 1 ccparams --set -f BCM957608-P2200GQF00_congestion_control_20250210_025627_19427.CFG
```

debug Command

The `debug` command dumps the device's internal configuration registers. The dump file can be used by Broadcom Customer Support for hardware troubleshooting.

Note: Operating System Support:

- This command is supported on Linux, Windows, FreeBSD, and VMWare operating systems.

Syntax

```
debug -c|--coredump [-d|--ddr] [-l|--l1cc] [-f <file name>]
debug -s|--snapdump [-f <file name>]
```

Parameters

`-c|--coredump`

Retrieves the register dump and crashdump information from the firmware.

`-s|--snapdump`

Retrieves the register dump, crashdump, firmware log, driver logs, and host tool logs.

`-d|--ddr`

Retrieves crashdump from the DDR if available and this option is only applicable to coredump command.

`-l|--l1cc`

Retrieves the context l1 cache and this option is only applicable to coredump command.

`-f`

File name to dump the coredump/snapdump. This is an optional parameter.

Examples

To collect the coredump:

```
niccli -i 1 debug -c
niccli -i 1 debug --coredump
niccli -i 1 debug --coredump -f test.core
```

To collect the coredump with L1 context cache:

```
niccli -i 1 debug --coredump --l1cc
```

To collect the snapdump:

```
niccli -i 1 debug -s
niccli -i 1 debug --snapdump
```

show Command

The `show` command is used to display all of the basic details of the device.

Note: Operating System Support:

- The`pcb_gen2_otp` option is supported only on Linux operating systems.
- The recommendations options are supported only on Linux, Windows, VMWare, and FreeBSD operating systems..
- The`list_ethernet` command is supported only on Linux, Windows, VMWare, and FreeBSD operating systems..

Syntax

```
show [-d | --device_info]
show -p | --pkg_ver [-f <firmware package file(s)>]
show -D | --device_pci_ids [-f <firmware package file(s)>]
show -c | --certificate [<-s | --slot number>]
show -n | --nvm_measurement
show --all
show --health
show -g | --pcb_gen2_otp
show -r | --recommendations [--pcie] [-V]
```

Parameters

-p | --pkg_ver

Display firmware package version installed on the device or in the package file.

-f

Displays the firmware package file information.

-D | --device_pci_ids

Displays the Broadcom device id information.

-d | --device_info

Displays the basic details of the device.

--all

Displays all the details of the device.

-c | --certificate

Displays the imported certificate chain on the device. This command is supported on BCM95750X and BCM957608 devices.

-s | --slot number

Slot number is where the certificate chain on the device is imported. The valid values are from 0 to 7. The default value is 0.

-n | --nvm_measurement

Displays whether the NVM configuration is active in the system. Has been changed or not. To facilitate this, a hash is generated based on nvm configuration. The hash represents the measurement of the configuration. This command is supported on BCM95750X and BCM957608 devices.

--health

Display the device health.

-g | --pcb_gen2_otp

Displays the PCB gen2 device OTP. This command is supported on BCM9574XX devices.

Examples

To display firmware package version installed on the device or in the package file:

```
niccli -i 1 show -p -f BCM957508-N2100G.pkg
niccli -i 1 show --pkg_ver
```

To display the basic details of the device:

```
niccli -i 1 show -d
niccli -i 1 show --device_info
```

To display the device health of the interface:

```
niccli -i 1 show --health
```

To display the imported certificate chain on the device:

```
niccli -i 1 show -c -s 1
niccli -i 1 show --certificate --slot 0
```

To display whether the NVM configuration that is active in the system:

```
niccli -i 1 show -n
niccli -i 1 show --nvm_measurement
```

To display Broadcom device id information:

```
niccli -i 1 show -D -f BCM957608-P2200GQF00.pkg
niccli -i 1 show --device_pci_ids -f BCM957608-P2200GQF00.pkg
```

To display all the details of the device:

```
niccli -i 1 show --all
```

To display the recommendation for self debug:

```
niccli -i 1 show -r
niccli -i 1 show --pcb_gen2_otp
niccli -i 1 show --recommendations --pcie
niccli -i 1 show --recommendations -V
```

To display the PCB gen2 device OTP:

```
niccli -i 1 show -g
niccli -i 1 show --recommendations
```

pcie Command

The `pcie` command is used to query and configure PCIe operations.

Note: Operating System Support:

- This command is supported on Linux, Windows, VMWare, and FreeBSD operating systems.

Syntax

```
pcie <-c|--compliance> --state <0/1>
pcie <-c|--compliance> --show
```

Parameters

-c|--compliance

Queries or configures the PCIe compliance operations..

--state

Configures the PCIe compliance state to enable or disable.

--show

Queries the PCIe compliance state.

Examples

To query the PCIe compliance state:

```
niccli -i 1 pcie --compliance --show
PCIe Compliance : Disabled
```

To enable or disable the PCIe compliance state:

```
niccli -i 1 pcie --compliance --state 0
niccli -i 1 show --health

PCIe compliance disabled successfully.
A system reboot is needed for the PCIe compliance to take effect.
```

VMWare Signed and Unsigned Command Mapping

Provides a list of mapped commands for VMWare unsigned and signed NICCLI commands.

From VMWare version ESXi8.0 and onwards, NICCLI is signed to support the VMWare plugin standards in secure and non-secure mode. The following table provides a map of NICCLI commands.

Table 56: VMWare Signed and Unsigned Command Mapping

Category	Unsigned NICCLI Commands	signed NICCLI Commands
Generic commands	niccli -v|--version	esxcli niccli version
	niccli -l|--list	esxcli niccli list
	niccli -d|--list_devices	esxcli niccli listdevices
	niccli -e|--list_ethernet	esxcli niccli listethernet
	niccli devid	esxcli niccli devid
config	niccli -i <index> nvm --view [-V] [-f <firmware package file name>] [-t|--type <nvm directory name>]	esxcli niccli nvm -c i -v <index> --view [-y|--verbosity] [-f|--filename=<str>] [-t|--type=<str>]
	niccli -i <index> nvm l|--list [-V] [-f <firmware package file name>]	esxcli niccli nvm -c i -v <index> -i|--list [-y|--verbosity] [-f|--filename=<str>]

Category	Unsigned NICCLI Commands	signed NICCLI Commands
	niccli -i <index> nvm --verify [-V] [-f <firmware package file name>]	esxcli niccli nvm -c i -v <index> --verify [-y|--verbosity] [-f|--filename=<str>]
	niccli -i <index> nvm -n|--sync	esxcli niccli nvm -c i -v <index> -n|--sync
	niccli -i <index> nvm -F|--restore_factory_defaults [--silent]	esxcli niccli nvm -c i -v <index> -F|--restore-factory-defaults
	niccli -i <index> nvm -r|--dir_read -f <file name> -t|--type <nvm directory name>	esxcli niccli nvm -c i -v <index> -r|--dir-read -f|--filename=<str> -t|--type=<str>
	niccli -i <index> nvm w|--dir_write -f <file name> -t|--type <nvm directory name>	esxcli niccli nvm -c i -v <index> -w|--dir-write -f|--filename=<str> -t|--type=<str>
	niccli -i <index> nvm -S|--saveoptions -f <file name>	esxcli niccli nvm -c i -v <index> -d|--saveoptions -f|--filename=<str>
	niccli -i <index> nvm -O|--optionhelp <option name>	esxcli niccli nvm -c i -v <index> -o|--optionhelp=<str>
	niccli -i <index> nvm -g|--getoption <option name> [--scope <scope index>]	esxcli niccli nvm -c i -v <index> -g|--getoption -N|--optionname=<str> [--scope=<str>]
	niccli -i <index> nvm -s|--setoption <option names with comma seperated> -v|--value <option value with comma seperated> [--scope <scope index>]	esxcli niccli nvm -c i -v <index> -s|--setoption=<str> -u|--value=<str> [--scope=<str>]
	niccli -i <index> nvm -b|--backup [--cfg]	esxcli niccli nvm -c i -v <index> -b|--backup [--cfg]
	niccli -i <index> nvm -L|--listoptions --diff	esxcli niccli nvm -c i -v <index> -a|--listoptions --diff
fwmanager	niccli -i <index> fw <-u|--update> -f <package file> [--force] [-y|--yes]	esxcli niccli fw -c i -v <index> -u|--update -f|--packagefile=<str> [--force] [-y|--yes]
	niccli -i <index> fw --reset	esxcli niccli fw -c i -v <index> --reset
	niccli -i <index> fw <-l|--livepatch>	esxcli niccli fw -c i -v <index> -p|--livepatch
	<--show>	--show
	niccli -i <index> fw <-l|--livepatch> <-a|--activate> [target_fw]	esxcli niccli fw -c i -v <index> -p|--livepatch -a|--activate [-w|--targetfw=<str>]

Category	Unsigned NICCLI Commands	signed NICCLI Commands
	niccli -i <index> fw <-l|--livepatch> <-d|--deactivate> [target_fw]	esxcli niccli fw -c i -v <index> -p|--livepatch -d|--deactivate [-w|--targetfw=<str>]
	niccli -i <index> fw <-l|--livepatch> <-p|--patch_update> [target_fw] -f <patch file>	esxcli niccli fw -c i -v <index> -p|--livepatch -t|--patchupdate [-w|--targetfw=<str>] -f|--packagefile=<str>
qos	niccli -i <index> qos <-n|--ingress> --cosq --show [-p|--persistent]	esxcli niccli qos -c i -v <index> -i|--ingress --cosq --show [-p|--persistent]
	niccli -i <index> qos <-n|--ingress> --cosq --set --state <value> [--mode <value>] [-p|--persistent]	esxcli niccli qos -c i -v <index> -i|--ingress --cosq --set --state=<str> [--mode=<str>] [-p|--persistent]
	niccli -i <index> qos <-e|--egress> --cosq --show [-p|--persistent]	esxcli niccli qos -c i -v <index> -r|--egress --cosq --show [-p|--persistent]
	niccli -i <index> qos <-e|--egress> --cosq --set --state <value> [-p|--persistent]	esxcli niccli qos -c i -v <index> -r|--egress --cosq --set --state=<str> [-p|--persistent]
	niccli -i <index> qos -g|--timed_tx_pacing_rate_profile --show	esxcli niccli qos -c i -v <index> -g|--timed-tx-pacing-rate-profile --show
linkdiag	niccli -i <index> linkdiag -T|--txfir --show <-M|--modulation_type> <mod_type> <-l|--lane> <lane_number>	esxcli niccli linkdiag -c i -v <index> -T|--txfir --show -M|--modulation-type=<str> -n|--lane=<lane_number>
	niccli -i <index> linkdiag -T|--txfir --set <-M|--modulation_type> <mod_type> <-l|--lane> <lane_mask> --pre1 <value> --pre2 <value> [--pre3 <value>] --main <value> --post1 <value> --post2 <value> [--post3 <value> --amp <value> --nlcl <value> --nlcu <value>]	esxcli niccli linkdiag -c i -v <index> -T|--txfir --set -M|--modulation-type=<str> -n|--lane=<lane_mask> --pre1=<str> --pre2=<str> [--pre3=<str>] --main=<str> --post1=<str> --post2=<str> [--post3=<str> --amp=<str> --nlcl=<str> --nlcu=<str>]
	niccli -i <index> linkdiag -F|--fdrstat [--start] [--stop] [--clear] [--counters]	esxcli niccli linkdiag -c i -v <index> -F|--fdrstat [--start] [--stop] [--clear] [--counters]
	niccli -i <index> linkdiag -D|--dscdump -l|--lane <lane_number> [-a|--diag_level <level>]	esxcli niccli linkdiag -c i -v <index> -D|--dscdump -n|--lane=<lane_number> [-a|--diag-level=<level>]
cable	niccli -i <index> cable -m|--module_info --show	esxcli niccli cable -c i -v <index> -m|--module-info --show

Category	Unsigned NICCLI Commands	signed NICCLI Commands
	niccli -i <index> cable -r|--read_module_eeprom [-p|--page_number <page number> -o|--offset <byte offset> -l|--length <number of bytes> -b|--bank <bank number> -i|--i2c_address <i2c addr>]	esxcli niccli cable -c i -v <index> -r|--read-module-eeprom -p|--page-number=<str> -o|--offset=<str> -n|--length=<str> b|--bank=<str> -i|--i2c-address=<str>
	niccli -i <index> cable -w|--write_module_eeprom -p|--page_number <page number> -o|--offset <byte offset> -v|--value <bytes>	esxcli niccli cable -c i -v <index> -w|--write-module-eeprom -p|--page-number=<str> -o|--offset=<str> -a|--value=<str>
	niccli -i <index> cable -M|--module_loopback -t|--loopback_type <type> [--lane <module_lane_mask>]	esxcli niccli cable -c i -v <index> -M|--module-loopback -t|--loopback-type=<type> [--lane=<str>]
link	niccli -i <index> link -s|--status	esxcli niccli link -c i -v <index> -s|--status
	niccli -i <index> link -c|--counters --show	esxcli niccli link -c i -v <index> -o|--counters --show
	niccli -i <index> link -p|--port_state <port state value>	esxcli niccli link -c i -v <index> -p|--portstate=<port state value>
	niccli -i <index> link -S|--port_speed [--speed <value>] [--lanes <number_of_lanes>] -a|--autoneg <on/off> [--fec <value>] [-t|--training <on/off>]	esxcli niccli link -c i -v <index> -S|--port-speed [--speed=<str>] [--lanes=<str>] -a|--autoneg=<str> [--fec=<str>] [-t|--training=<str>]
timesync	niccli -i <index> timesync <-d | --dutycycle> <--period> <value> --up <value>	esxcli niccli timesync -c i -v <index> -d|--dutycycle --period=<str> --up=<str>
	niccli -i <index> timesync <--dll> <-s|--source> <value> <-q|--frequency> <value>	esxcli niccli timesync -c i -v <index> --dll -o|--source=<value> -q|--frequency=<value>
	niccli -i <index> timesync <--ptp> --show	esxcli niccli timesync -c i -v <index> -p|--ptp --show
	niccli -i <index> timesync <--ptp> --set <-p | --primary_pf> <pid> [<-v | --primary_vf> <vfid>] [<-P | --secondary_pf> <pfid>] [<-V | --secondary_vf> <vfid>]	esxcli niccli timesync -c i -v <index> -p|--ptp --set -r|--primarypf=<str> [-x|--primaryvf=<str>] [-z|--secondarypf=<str> -w|--secondaryvf=<str>]
	niccli -i <index> timesync <--synce> --show	esxcli niccli timesync -c i -v <index> -y|--synce --show

Category	Unsigned NICCLI Commands	signed NICCLI Commands
	niccli -i <index> timesync <--synce> --set <-Q | --frequency_profile> <value> [<-c|--primary_clock_state> <value>] [<-C|--secondary_clock_state> <value>]	esxcli niccli timesync -c i -v <index> -y|--synce --set -b|--frequency-profile=<str> [-u|--primary-clock-state=<str>] [-C|--secondary-clock-state=<str>]
	niccli -i <index> timesync <--tsio> --show	esxcli niccli timesync -c i -v <index> --tsio --show
	niccli -i <index> timesync <--tsio> <-t | --tsio_function_pin> <idx> <-u | --pin_usage_string> <value> <--state> <value>	esxcli niccli timesync -c i -v <index> --tsio -t|--tsio-function-pin=<str> -k|--pin-usage-string=<str> --state=<str>
counters	niccli -i <index> counters -p | --pcie [-V]	esxcli niccli counters -c i -v <index> -p|--pcie [-V|--verbosity]
	niccli -i <index> counters -c | --clear	esxcli niccli counters -c i -v <index> -a|--clear
tunnel	niccli -i <index> show [-d | --device_info]	esxcli niccli show -c i -v <index> [-i|--device-info]
	niccli -i <index> show -p | --pkg_ver [-f <firmware package file(s)>]	esxcli niccli show -c i -v <index> -p|--pkg-ver [-f|--filename=<str>]
	niccli -i <index> show -D | --device_pci_ids [-f <firmware package file(s)>]	esxcli niccli show -c i -v <index> -d|--device-pci-ids [-f|--filename=<str>]
	niccli -i <index> show -c | --certificate [<-s | --slot number>]	esxcli niccli show -c i -v <index> -R|--certificate [s|--slot=<value>]
	niccli -i <index> show -n | --nvm_measurement	esxcli niccli show -c i -v <index> -n|--nvm-measurement
	niccli -i <index> show --all	esxcli niccli show -c i -v <index> --all
	niccli -i <index> show --health	esxcli niccli show -c i -v <index> --health
	niccli -i <index> show -r | --recommendations [--pcie] [-V]	esxcli niccli show -c i -v <index> -r|--recommendations [--pcie] [-V|--verboselevel]
ring resource configuration	niccli -i <index> resmgmt [--pf/--all] [<-p|--profile> | < --min > |< --max > | <-r | --roce_max> | < -m | --max_completion_rings> | < -s | --strategy >] --show	esxcli niccli resmgmt -c i -v <index> [--pf|--all] [<-p|--profile> | --min | --max | <-r|--roce-max> | <m|--max-completion-rings> | <-s|--strategy>] --show

Category	Unsigned NICCLI Commands	signed NICCLI Commands
	niccli -i <index> resmgmt [--pf/--all] --set [<-p|--profile> | <--min> |<--max> | <-r|--roce_max> | <-m|--max_completion_rings>] [< --bw <bandwith of each pf with comma separated>] [< -s | --strategy > < minimal/maximal/minimal-static>]	esxcli niccli resmgmt -c i -v <index> [--pf|--all] --set [<-p|--profile> | --min | --max | <-r|--roce-max> | <m|--max-completion-rings>] [-b|--bw=<str>] [<-s|--strategy> -y|--strategytoset=<str>]
Congestion Control Configuration	niccli -i <index> ccparams -d|--dump	esxcli niccli ccparams -c i -v <index> -d|--dump
	niccli -i <index> ccparams --set -f <configuration file>	esxcli niccli ccparams -c i -v <index> --set -f|--configfile=<str>
debug	niccli -i <index> debug -c|--coredump [-l|--l1cc] [-f <file name>]	esxcli niccli debug -c i -v <index> -r|--coredump [-L|--l1cc] [-f|--filename=<str>]
	niccli -i <index> debug -s|--snapdump [-f <file name>]	esxcli niccli debug -c i -v <index> -s|--snapdump [-f|--filename=<str>]
PCIe operations command	niccli -i <index> pcie <-c|--compliance> --state <0/1>	esxcli niccli pcie -c i -v <index> -C|--compliance --state=<str>
	niccli -i <index> pcie <-c|--compliance> --show	esxcli niccli pcie -c i -v <index> -C|--compliance --show

Mapping NICCLI Commands (233 to 234 and Later Versions)

Provides a list of mapped commands for NICCLI for 233 to 234 and later versions.

The following table maps the NICCLI 233 commands to the NICCLI 234 and later commands.

Table 57: NICCLI 233 to 234 and Later Command Mapping

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
Generic	Lists all of the compatible devices.	list|--list	-l|--list
	Command Example:	niccli --list	niccli --list
Generic	Lists all of the supported devices with basic information.	--listdev	-d|--list_devices
	Command Example:	niccli --listdev	niccli -d
Generic	Query Broadcom device ID's.	devid	devid
	Command Example:	niccli devid	niccli devid
Generic	Display FW PKG version installed on the device.	pkgver	--nvm_measurement

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
	Command Example:	`niccli pkgver`	`niccli -i <index> show --pkg_ver`
Generic	Verify FW packages and NVM.	`verify`	`verify`
	Command Example:	`niccli verify`	`niccli verify`
Generic	Lists all NIC devices with ethernet interface names.	`list-eth|--list-eth`	`-e|--list_ethernet`
	Command Example:	`niccli list-eth`	`niccli -e`
Generic	Lists the available commands.	`help`	`-h|--help`
	Command Example:	`niccli help`	`niccli -h`
Generic	Quits from the application (Applicable in interactive mode only).	`quit`	`quit`
	Command Example:	`niccli -i 1 quit`	`niccli -i 1 quit`
Generic	Displays the current version of the application.	`version`	`-v|--version`
	Command Example:	`niccli -i 1 version`	`niccli -v`
Show	Shows NIC specific device information.	`show`	`show`
	Command Example:	`niccli -i 1 show`	`niccli -i 1 show -d`
Debug	Retrieves coredump data from device.	`coredump`	`debug -c|--coredump`
	Command Example:	`niccli -i 1 coredump`	`niccli -i 1 debug -c`
Debug	Retrieves snapdump data from device.	`snapdump`	`debug -s|--snapdump`
	Command Example:	`niccli -i 1 snapdump`	`niccli -i 1 debug -s`
fw	Install/Update FW.	`install`	`fw -u|--update`
	Command Example:	`niccli -i 1 install BCM957508-P2100G.pkg`	`niccli -i 1 fw --update -f BCM957608-P1400GDF00.pkg`
fw	Reset the device.	`reset`	`fw --reset`
	Command Example:	`niccli -i 1 reset`	`niccli -i 1 fw --reset`
fw	Query, Activate, and Deactivate the patch.	`livepatch`	`fw -l|--livepatch`
	Command Example:	`niccli -i 1 livepatch`	`niccli -i 1 fw --livepatch --show`
nvm	Displays NVM components and the associated versions.	`nvmlist`	`nvm -l|--list`
	Command Example:	`niccli nvmlist`	`niccli nvm -l`
nvm	View NVM directories data.	`nvmview`	`nvm --view`

Category	Commands (All command examples are run with sudo)	NICCLI 233 Command	NICCLI 234/Later Command
	Command Example:	`niccli nvmview`	`niccli nvm --view`
nvm	NVRAM Option Management.	`nvm`	`nvm`
	Command Example:	`niccli -i 1 nvm -getoption`	`niccli -i 1 nvm -g`
nvm	Restores NVM configuration to factory defaults.	`rfd`	`nvm -F|--restore_factory_defaults`
	Command Example:	`niccli -i 1 rfd`	`niccli -i 1 nvm --restore_factory_defaults`
nvm	Synchronize primary and secondary FW images.	`fw_sync`	`nvm -n|--sync`
	Command Example:	`niccli -i 1 fw_sync`	`niccli -i 1 nvm -n`
nvm	Create or overwrite NVM data item with a file.	`write`	`nvm -w|--dir_write`
	Command Example:	`niccli -i 1 write -f data.txt -type pkglog`	`niccli -i 1 nvm -w -f data.txt -t pkglog`
nvm	Read the NVM item data and write its contents to a file.	`read`	`nvm -r|--dir_read`
	Command Example:	`niccli -i 1 read -type pkglog -f data.txt`	`niccli -i 1 nvm -r -t pkglog -f data.txt`
link	Query or configure the port state.	`portstate`	`link -p|--port_state`
	Command Example:	`niccli -i 1 portstate -get`	`niccli -i 1 link -s`
linkdiag	Network Interface Card Transmission Finite Impulse Response.	`txfir`	`linkdiag -T|--txfir`
	Command Example:	`niccli -i 1 txfir -get -modtype PAM4 -lane 0`	`niccli -i 1 linkdiag -T --show --modulation_type PAM4 --lane 0`
linkdiag	Run PRBS loopback test.	`prbs_test`	`linkdiag -P|--prbs_test`
	Command Example:	`niccli -i 1 prbs_test -enable`	`niccli -i 1 linkdiag --prbs_test --enable`
linkdiag	Query/perform loopback config.	`loopback`	`linkdiag -L|--loopback`
	Command Example:	`niccli -i 1 loopback`	`niccli -i 1 linkdiag --loopback --show`
linkdiag	Retrieves dscdump for device.	`dscdump`	`linkdiag -D|--dscdump`
	Command Example:	`niccli -i 1 dscdump -lane 0`	`niccli -i 1 linkdiag -D --lane 0`

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
serdes	Plots the SerDes PCI and ethernet eye and prints the horizontal and vertical margin values.	`serdes`	`serdes`
	Command Example:	`niccli -i 1 serdes -pci -lane 0 -plot`	`niccli -i 1 serdes --eye --pci --lane 0 --plot`
counters	Show/Execute pcie operation.	`pcie`	`counters -p | --pcie`
	Command Example:	`niccli -i 1 pcie -counters`	`niccli -i 1 counters -p`
counters	Clear the port counters.	`clearcounters`	`counters -c | --clear`
	Command Example:	`niccli -i 1 clearcounters`	`niccli -i 1 counters -c`
mh	Configure and Query for the number of PFs per PCIe endpoint.	`pfalloc`	`mh -p|--pf_alloc`
	Command Example:	`niccli -i 1 pfalloc -get`	`niccli -i 1 mh --pf_alloc --show`
msix	Displays and configures the number of MSIX max vectors values for VF's per each PF.	`msixmv`	`msix -m|--max_vectors`
	Command Example:	`niccli -i 1 msixmv -get -all`	`niccli -i 1 msix --max_vectors --show --all`
qos	Queries and configures the ingressqos parameters.	ingressqos	qos -n|--ingress
	Command Example:	`niccli -i 1 ingressqos -cosq -get`	`niccli -i 1 qos --ingress --cosq --show`
qos	Queries and configures the egressqos parameters.	`egressqos`	`qos -e|--egress`
	Command Example:	`niccli -i 1 egressqos -cosq -get`	`niccli -i 1 qos --egress --cosq --show`
qos	Configures the priority-based flow control for a given priority.	`pfc`	`qos --pfc`
	Command Example:	`niccli -i 1 pfc -enable 5,6`	`niccli -i 1 qos --pfc --enable 5,6`
qos	Configures the priority for the AppTLV.	`apptlv`	`qos --apptlv`
	Command Example:	`niccli -i 1 apptlv -add -app 5,1,35093`	`niccli -i 1 qos --apptlv --add --app 5,1,35093`
qos	Configures the rate limit for each traffic class.	`tcrlmt`	`qos --tc`

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
	Command Example:	`niccli -i 1 tcrlmt -set -r 10`	`niccli -i 1 qos --tc --set --rate_limit 10`
qos	Configures the enhanced transmission selection, priority to traffic class, and bandwidths	`ets`	`qos -E|--ets`
	Command Example:	`niccli -i 1 ets -tsa 0:ets,1:ets,2:strict -up2tc 0:0,1:0,2:0 -tcbw 70,30`	`niccli -i 1 qos --ets --tsa 0:ets,1:ets,2:strict --up2tc 0:0,1:0,2:0 --tcbw 70,30`
qos	Configures the user priorities to traffic classes.	`up2tc`	`qos --up2tc`
	Command Example:	`niccli -i 1 up2tc -p 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0`	`niccli -i 1 qos --up2tc 0:0,1:0,2:0,3:0,4:0,5:1,6:0,`
qos	Queries the configured enhanced transmission selection, priority to traffic class and bandwidths.	`getqos`	`qos -E|--ets`
	Command Example:	`niccli -i 1 getqos`	`niccli -i 1 qos --ets --show`
qos	Lists the priority to traffic class and queueid mapping.	listmap	`qos -l|--listmap`
	Command Example:	`niccli -i 1 listmap -pri2cos`	`niccli -i 1 qos --listmap --pri2cos`
qos	Queries the dscp to priority mapping.	`dscp2prio`	`qos -D|--dscp2prio`
	Command Example:	`niccli -i 1 dscp2prio`	`niccli -i 1 qos --dscp2prio`
qos	Configures the receive side port rate limit.	`rxportrlmt`	`qos -r|--rx_port_rate_limit`
	Command Example:	`niccli -i 1 rxportrlmt -set -max 40`	`niccli -i 1 qos --rx_port_rate_limit --set --max 40`
qos	Queries the configured receive side rate control parameters.	`rxrlmt`	`qos -R|--rx_rate_limit`
	Command Example:	`niccli -i 1 rxrlmt -get`	`niccli -i 1 qos --rx_rate_limit --show`
qos	Configures the receive side rate control parameters for a given endpoint.	`rxeprlmt`	`qos -X|--rx_ep_rate_limit`
	Command Example:	`niccli -i 1 rxeprlmt -set -ep0_max 0`	`niccli -i 1 qos --rx_ep_rate_limit --set --ep0 0`

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
qos	Queries and Configures the transmit side partition rate limit applies to traffic sent from a partition, which is one PF and all of its child VFs.	`txpartitionrlmt`	`qos -t|--tx_partition_rate_limit`
	Command Example:	`niccli -i 1 txpartitionrlmt -get`	`niccli -i 1 qos --tx_partition_rate_limit --show`
qos	Queries and Configures the transmit side of port rate limit.	`txportrlmt`	`qos -P|--tx_port_rate_limit`
	Command Example:	`niccli -i 1 txportrlmt -get`	`niccli -i 1 qos --tx_port_rate_limit --show`
qos	Queries and Configures the PCIe endpoint.	`txeprlmt`	`qos -x|--tx_ep_rate_limit`
	Command Example:	`niccli -i 1 txeprlmt -set -p 0 -ep0_max 50`	`niccli -i 1 qos --tx_ep_rate_limit --set --port 0 --ep0 50`
timesync	TSIO function capability on the pin.	`tsio`	`timesync --tsio`
	Command Example:	`niccli -i 1 tsio`	`niccli -i 1 timesync --tsio -t 3 -u 1 --state 1`
timesync	Sets duty cycle on TSIO outgoing signal.	`dutycycle`	`timesync -d|--dutycycle`
	Command Example:	`niccli -i 1 dutycycle -period 1 -up 0`	`niccli -i 1 timesync --dutycycle --period 1 --up 0`
timesync	Sets the DLL source for PHC.	`dllsource`	`timesync --dll`
	Command Example:	`niccli -i 1 dllsource -source 1 -frequency 1`	`niccli -i 1 timesync --dll --source 1 --frequency 3`
timesync	PTP extended parameters operation.	`ptp`	`timesync -p| --ptp`
	Command Example:	`niccli -i 1 ptp -get`	`niccli -i 1 timesync --ptp --show`
timesync	Configures the synchronous ethernet profile.	`synce`	`timesync --synce`
	Command Example:	`niccli -i 1 synce -get`	`niccli -i 1 timesync --synce --show`
vf	Configures and Queries for a trusted VF.	`vf`	`vf`

Category	Commands (All command examples are run with `sudo`)	NICCLI 233 Command	NICCLI 234/Later Command
	Command Example:	`niccli -i 1 vf -id 1 -trust`	`niccli -i 1 vf --trust --show --vf_index 1`
vf	Adds ntuple flow filter.	`add_ntuple_filter`	`vf --add_ntuple_filter`
	Command Example:	`niccli -i 1 add_ntuple_filter 00:01:02:03:04:a3 1023 0xFFFF 0 1`	`niccli -i 1 vf --add_ntuple_filter --macaddress 00:01:02:03:04:a3 -p 1023 -P 0xFFFF -v 1 -T 1`
vf	This command is used to configure the GPU host and guest physical address mapping.	`peermemmap`	`vf -M|--peer_mem_map`
	Command Example:	`niccli -i 1 peermemmap -hpa 0x1FFFFFFF,0x2FFFFFFF,0x3FFFFFFF -gpa 0x9FFFFFFF,0xAFFFFFFF,0xBFFFFFFF -size 0x10000,0x10000,0x8000,0x8000`	`niccli -i 1 vf --peer_mem_map --hpa 0x1FFFFFFF,0x2FFFFFFF,0x3FFFFFFF --gpa 0x9FFFFFFF,0xAFFFFFFF,0xBFFFFFFF --size 0x10000,0x10000,0x8000,0x8000`

NIC Apps

Provides information on the NIC Apps utility.

NIC Apps is an integrated tool, which provides a single interface to multiple tools or scripts which are used to help detect and resolve issues. NIC Apps commands are executed on the server system which has a Broadcom network adapter installed.

The following utility is provided as part of NIC Apps:

- System Status Check Utility

NIC Apps Syntax

All NIC App commands used the same syntax. The following is the standard syntax structure of an NIC App command:

```
bcm_nicapps.py <namespace> [<namespace> ...] <cmd> [cmd options]
```

Table 58: Command Structure Descriptions

Syntax Element	Description
namespace	The Namespace element allows NIC Apps commands to be grouped. For example, commands such as `syscheck` and `biosdump` can be part of same namespace element `diag` . Namespace can also have sub namespaces.
cmd	The command used to run the integrated tool or scripts. For example, this could be SOS or Tune.
options	The options element are options that can be supplied to commands. For example, NIC Apps config tune can have options `-r` or `-t` .

System Status Check Utility

Provides information on the system status check utility.

Introduction

The system status check utility has the following requirements:

- The utility requires that the target system is configured with Broadcom L2 and RoCE drivers for a RoCE enabled environment.
- The NICCLI tool must be installed, and in the user `$PATH` environment variable.

If all checks pass, a `System appears correctly configured.` message is displayed.

Syntax

```
bcm_nicapps.py diag syscheck
```

Features

The system check utility provides the following checks:

- The following NVM options are checked: `performance_profile: RoCE` , `enable_udcc: 0` , `pci_relaxed_ordering: Enabled`
- The system memory is checked for equal configuration across processors and channels.
- PFC-enabled queues are checked to ensure the queue lossless traffic flag is set.
- The RoCE QoS settings are checked for correctness.
- A separate network for each supported, enabled device.

NIC Tune

Provides information on automatic tuning for Linux systems using the NIC Tune utility.

Quickstart

The `nictune.py` tool reports status on a list of tunable parameters of all supported network interface cards when invoked as:

```
nictune.py
```

To optimally tune all supported network adapters for the default maximum throughput profile, and install scripts to persist those changes between reboots, invoke as:

```
nictune.py -t -i
```

See `nictune.py --help` output for all available options.

Introduction

This tool automates performance tuning for Broadcom Ethernet Network Interface Cards (NICs). The following tuning parameters are managed:

- NIC configured performance profile
- Ethtool TX/RX/combined queue sizes
- Hardware generic receive offload (GRO)
- Interrupt CPU affinity (IRQ Affinity)
- Kernel IOMMU
- Kernel performance governor
- TCP buffer memory
- Transmit packet steering (XPS)
- Receive Flow Steering (RFS)
- Interrupt Coalescing

The following parameters are managed on RoCE profile:

- NIC PCIe Relaxed Ordering enablement
- NIC User Defined Congestion Control
- RoCE DCSP TLV configured
- CNP DCSP TLV configured
- CoS Queue PFC enabled Lossless

The tool has the following modes:

- Report: Without arguments, the tool reports status of each tuning parameter for each supported NIC in the system.
- Tune: The `-t` argument applies optimal values to the running system, but these are reset after any reboot.
- Install: The `-i` argument installs boot-time scripts to configure the tuning parameters, making the change permanent.
- Uninstall: The `-u` argument removes the installed scripts from any previous run with the `-i` argument.
- Performance Benchmarking with iperf3: The `-P` argument provides support for iperf3 performance benchmarking.

Profiles

The tool supports two types of profiles for the different parameters when invoked by the `-p` argument:

- Throughput: A default profile that provides the parameters that are focused on L2 performance tuning.
- RoCE: `-p roce` provides parameters that are related to RoCE features and only supports reporting mode.

Performance Test Mode

The tool supports performance testing by `iperf` or `iperf3` . The`--runperf` argument runs the performance test.

- Client mode: The `--perfclient` argument runs the performance client, connecting to the specified IP address.
- Server mode: The `--perfserver` argument runs the performance server process.

Note: The `-P` or `--parallel` option is available to enable concurrent processing when using the `iperf3` tool. This option has no effect when used with the `iperf` tool.

Multiple IP addresses can be assigned with multiple ports at the same time, or you can select a specific device with `-d` .

The result will show at the end of output for each devices, as shown below:

```
[RESULT] Performance result on eth0: xxx Gbit/s
[RESULT] Performance result on eth1: xxx Gbit/s
```

Examples

System-A and System-B have installed N2200GDP (2x200G) adapters and the interface names are eth0 and eth1.

```
In system-A, the IP address of eth0 is 192.168.1.1 and the IP address of eth1 is 192.168.2.1
In system-B, the IP address of eth0 is 192.168.1.2 and the IP address of eth1 is 192.168.2.2
```

Case 1: Run uni-directional traffic on both ports between System-A and System-B after applying optimal tuning values `(-t)`.

```
System-A with server mode: ./nictune.py -t --runperf --perfserver
System-B with client mode: ./nictune.py -t --runperf --perfclient 192.168.1.1,192.168.2.1
```

Case 2: Run bi-directional traffic on both ports between System-A and System-B after applying optimal tuning values `(-t)`.

```
System-A : ./nictune.py -t --runperf --perfserver --perfclient 192.168.1.2,192.168.2.2
System-B : ./nictune.py -t --runperf --perfserver --perfclient 192.168.1.1,192.168.2.1
```

Case 3: Run bi-directional traffic on eth0 only between System-A and System-B after applying optimal tuning values `(-t)`.

```
System-A : ./nictune.py -t -d eth0 --runperf --perfserver --perfclient 192.168.1.2
System-B : ./nictune.py -t -d eth0 --runperf --perfserver --perfclient 192.168.1.1
```

Requirements

Supported Operating Systems

See the Supported Operating Systems for Ethernet Network Adapters page for the full list of supported operating systems and platforms.

Supported Devices

The following Broadcom devices are fully supported by this tool:

- BCM95741X
- BCM95750X
- BCM957608

All speeds and cable/transceiver types supported by the above devices are supported by this tool.

Software Requirements

The Broadcom driver (`bnxt_en`) must be installed and configured. The Broadcom NICCLI tool must also be installed to manage some tuning parameters. The `iperf/iperf3` utility must be installed to run the performance testing. See Installing Software for Ethernet Network Adapters for additional information.

RoCE Analyzer

Provides information on the RoCE analyzer utility.

This tool runs diagnostics on RoCE configurations to verify the network interfaces, inspect configurations, run performance tests, collect and diagnose statistical counters in order to ensure correctness of RoCE configurations. The RoCE analyzer utility helps in preventing common misconfigurations and facilitates the identification and troubleshooting of RoCE failures. The RoCE Analyzer tool is compatible with RHEL, SLES, and Ubuntu operating systems. It offers

both short and long-form command formats. For a comprehensive list of available options, refer to the output of `roceanalyzer.py --help`.

The RoCE analyzer utility command details are provided in the following sections:

- Tool Command
- Software Command
- Performance Command
- BIOS Command
- Layer2 Connectivity Command
- Link Command
- NVM Command
- UDCC Command
- Congestion Control Commands
- QoS Command
- Recommendations Command
- RoCE Debug Command
- Statistics Command
- All Command

Tool Command

The `tool` command performs verifications on the pre-requisite RDMA tools.

Syntax

```
roceanalyzer.py <-t|--tool> [-j|--json [-f|--file <file name>]]
```

Parameters

`-t|--tool`

Performs pre-requisite RDMA tool verifications.

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -t
roceanalyzer.py -t -j
roceanalyzer.py -t -j -f <file name>
roceanalyzer.py --tool
roceanalyzer.py --tool --json
roceanalyzer.py --tool --json <file name>
```

Software Command

The `software` command performs verifications on the RoCE driver and firmware versions across all device interfaces.

Syntax

```
roceanalyzer.py <-s|--software> [-j|--json [-f|--file <file name>]]
```

Parameters

-s|--software

Performs RoCE Driver and firmware version verifications on network interface card.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -s
roceanalyzer.py -s -j
roceanalyzer.py -s -j -f <file name>
roceanalyzer.py --software
roceanalyzer.py --software --json
roceanalyzer.py --software --json <file name>
```

Performance Command

The `performance` command performs verifications on the host configurations for optimal performance.

Syntax

```
roceanalyzer.py <-p|--perf> [-j|--json [-f|--file <file name>]]
```

Parameters

-p|--perf

Performs host performance configuration verifications.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -p
roceanalyzer.py -p -j
roceanalyzer.py -p -j -f <file name>
roceanalyzer.py --perf
roceanalyzer.py --perf --json
roceanalyzer.py --perf --json <file name>
```

BIOS Command

To retrieve the host BIOS settings, the RoCE analyzer requires the BMC server IP, HTTP retry, and timeout values to be entered in the `bmcServerInfo` file.

Verification of BIOS configuration settings necessitates authentication. When prompted, provide the username and password to access the system's secure configuration interface.

The format of the `bmcServerInfo` file is as follows:

```
# Timeout in seconds, default is 3 seconds
TIMEOUT 3
# Default retries is 1 ( 1 original + 1 retry)
RETRIES 3
# BMC IP address. Replace xx.xx.xx.xx with your BMC IP.
BMC_IP xx.xx.xx.xx
```

See the system's BIOS console to ensure that the BIOS settings are correctly configured for the server, as these settings are system-specific. See BIOS Tuning on Ethernet Network Adapters for more information.

Syntax

```
roceanalyzer.py <-b|--bios> [-j|--json [-f|--file <file name>]]
```

Parameters

-b|--bios

Verifies the BIOS configuration settings.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -b
roceanalyzer.py -b -j
roceanalyzer.py -b -j -f <file name>
roceanalyzer.py --bios
roceanalyzer.py --bios --json
roceanalyzer.py --bios --json <file name>
```

Layer2 Connectivity Command

The `layer2` command performs verifications on the L2 connectivity of the network interfaces.

To do this, the server and client IP addresses and subnet mask (in CIDR notation) for the backend NIC interfaces must be entered into the `serverInfo` file. Users will be prompted for a username and password, though these credentials are ignored if the server is set up for passwordless authentication.

The `serverInfo` file uses the following format:

```
# Provide the subnet in CIDR notation.
# Example: subnet=192.168.1.0/24
# Provide the server IP address.
# Example: 10.136.14.1
```

```
# Provide the client IP address.
# Example: 10.136.14.2
subnet=192.168.1.0/24
10.136.123.1
10.136.123.2
```

Syntax

```
roceanalyzer.py <-l|--layer1> [-j|--json [-f|--file <file name>]]
```

Parameters

-l|--layer2

Performs L2 connectivity verifications on the network interfaces.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -l
roceanalyzer.py -l -j
roceanalyzer.py -l -j -f <file name>
roceanalyzer.py --layer2
roceanalyzer.py --layer2 --json
roceanalyzer.py --layer2 --json <file name>
```

Link Command

The `link` commands perform verifications on the link configurations. By default, operations run on all available device interfaces unless the `-d` option is specified.

Syntax

```
roceanalyzer.py <-L|--link> [-j|--json [-f|--file <file name>]]
```

Parameters

-L|--link

Performs link verifications on network interface card.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -L
roceanalyzer.py -L -j
roceanalyzer.py -L -j -f <file name>
roceanalyzer.py --link
```

```
roceanalyzer.py --link --json
roceanalyzer.py --link --json <file name>
```

NVM Command

The `nvm` command performs verifications on the NVM configurations required for RoCE.

Syntax

```
roceanalyzer.py <-n|--nvm> [-j|--json [-f|--file <file name>]]
```

Parameters

`-n|--nvm`

Performs NVM verifications on network interface card.

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -n
roceanalyzer.py -n -j
roceanalyzer.py -n -j -f <file name>
roceanalyzer.py --nvm
roceanalyzer.py --nvm --json
roceanalyzer.py --nvm --json <file name>
roceanalyzer.py --nvm --json <file name>
```

UDCC Command

The `udcc` commands perform verifications if the network interface card has udcc enabled and the card is udcc promoted. By default, without the `-d` option, these operations are performed on all available device interfaces.

Syntax

```
roceanalyzer.py <-u|--udcc> [-j|--json [-f|--file <file name>]]
```

Parameters

`-u|--udcc`

Performs udcc verification on network interface card.

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -u
```

```
roceanalyzer.py -u -j
roceanalyzer.py -u -j -f <file name>
roceanalyzer.py --udcc
roceanalyzer.py --udcc --json
roceanalyzer.py --udcc --json <file name>
```

Congestion Control Commands

The `cc` commands perform verification of congestion control configurations. These commands support 100G, 200G, and 400G congestion control parameter verification. By default, the operation runs on all available device interfaces if the `-d` option is not specified.

Syntax

```
roceanalyzer.py <-c|--cc> [-j|--json [-f|--file <file name>]]
```

Parameters

`-c|--cc`

Performs congestion control configuration verifications on network interface card.

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -c
roceanalyzer.py -c -j
roceanalyzer.py -c -j -f <file name>
roceanalyzer.py --cc
roceanalyzer.py --cc --json
roceanalyzer.py --cc --json <file name>
```

QoS Command

The `qos` commands perform verifications on the QoS configurations. The RoCE analyzer validates these configurations using parameters from `roceConfigFile` , however, only when OVERRIDE is set to 1. If OVERRIDE is not set, or if `roceConfigFile` is not present, the system defaults to the QoS settings found in `/usr/bin/bnxt_re_conf.sh` . By default, operations run on all available device interfaces unless the `-d` option is specified.

The `roceConfigFile` file uses the following format:

```
# This file defines the expected QoS configuration settings.
# By default the values are read from /usr/bin/bnxt_re_conf.sh.
# Set OVERRIDE to 1 to use the following values for QoS validation.
OVERRIDE=1
EXPECTED_FC_MODE=3
EXPECTED_ROCE_PRI=3
EXPECTED_ROCE_DSCP=26
EXPECTED_CNP_PRI=7
```

```
EXPECTED_CNP_DSCP=48
EXPECTED_ROCE_BW=50
```

Syntax

```
roceanalyzer.py <-q|--qos> [-j|--json [-f|--file <file name>]]
```

Parameters

-q|--qos

Performs QoS verifications on network interface cards.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -q
roceanalyzer.py -q -j
roceanalyzer.py -q -j -f <file name>
roceanalyzer.py --qos
roceanalyzer.py --qos --json
roceanalyzer.py --qos --json <file name>
```

Recommendations Command

The `recommendations` command performs verifications on the host configurations for optimal performance. Without the `-d` option, the operation will default to running on all available device interfaces.

Syntax

```
roceanalyzer.py <-r|--recommendations> [-j|--json [-f|--file <file name>]]
```

Parameters

-r|--recommendations

Displays the recommendations for troubleshooting.

-j|--json

This option provides the output in JSON format.

-f|--file

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -r
roceanalyzer.py -r -j
roceanalyzer.py -r -j -f <file name>
roceanalyzer.py --recommendations
roceanalyzer.py --recommendations --json
roceanalyzer.py --recommendations --json <file name>
```

Debug Telemetry Command

The `debug` command performs verifications on debug telemetry functionality via NICCLI and Ethtool. By default, the operation will run on all available device interfaces unless the `-d` option is specified.

Syntax

```
roceanalyzer.py <-D|--debugl> [-j|--json [-f|--file <file name>]]
```

Parameters

`-D|--debug`

Verifies debug telemetry functionality.

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -D
roceanalyzer.py -D -j
roceanalyzer.py -D -j -f <file name>
roceanalyzer.py --debug
roceanalyzer.py --debug --json
roceanalyzer.py --debug --json <file name>
```

Statistics Command

This option is used to assess RoCE performance and to validate RoCE configurations. It enables basic and extended RDMA connectivity tests and collect RoCE statistics. Before initiating RDMA connectivity tests, execute `./roceanalyzer.py -l` to confirm L2 connectivity. For extended RDMA connectivity tests, input the server and client IP addresses along with their subnet masks (in CIDR format) for the back-end NICs into the `serverInfo` file. You will be prompted to enter a username and password.

Note: The statistics command cannot be used in conjunction with the following options: -a, -u, -c, -L, -q, -p, -l, -t, -s, -n, -r, -D.

Syntax

To run basic RDMA connectivity tests on a single interface on server and client, provide the NIC device and interface connection details through the `-d` , `--device` and `--remote_ip` options.

To run RoCE Perf Tests in server mode:

```
roceanalyzer.py -S --basic --server -d <local_device>
```

To run RoCE Perf Tests in client mode as well as monitor RoCE statistics and measure performance:

```
roceanalyzer.py -S --basic --client -d <local_device>` --remote_ip <remote_nic_ip_address>
```

To view RoCE Statistics on the client, run the previous command with the -v, --verbose option.

To run extended RDMA connectivity tests on full mesh. Run RoCE Perf Tests in server mode:

```
roceanalyzer.py -S -e --server
```

To run RoCE Perf Tests in client mode as well as monitor RoCE statistics and measure performance:

```
roceanalyzer.py -S -e --client
```

To view RoCE Statistics on the client, run the previous command with the -v, --verbose option.

All Command

The `all` command option is used to run all of the above command options to perform verifications except certain on demand commands such as checking congestion control settings (-c), layer2 connectivity (-l), and statistics (-S).By default, the operations run on all available device interfaces.

Syntax

```
roceanalyzer.py <-a|--all> [-j|--json [-f|--file <file name>]]
```

Parameters

`-a|--all`

Runs all RoCE Analyzer commands

`-j|--json`

This option provides the output in JSON format.

`-f|--file`

This option redirects the output in JSON format into a file

Examples

```
roceanalyzer.py -a
roceanalyzer.py -a -j
roceanalyzer.py -a -j -f <file name>
roceanalyzer.py --all
roceanalyzer.py --all --json
roceanalyzer.py --all --json <file name>
```

When specifying network interface names, separate each name with a space. The command format is: `roceanalyzer.py -d [device1 device2 ...] [command option/s]` . This option is only effective when used with the following options: -L, -c, -n, -q, -u, -D, -r.

SOS Reporting Tool

Provides information on installing and using the Broadcom SOS reporting tool.

Introduction

The Broadcom SOS reporting tool builds on the open source bcm_sosreport tool to collect system information for support purposes. The following sections describe how to create the `bcm_sosreport` package and how to install and run the tool. The report that is generated can be sent to a Broadcom support representative for analysis.

Features

In addition to the default data collection plugins that come with `sosreport`, the package also includes the following:

- AMD and Nvidia GPU topology reports if clinfo or nvidia-smi utilities are installed.
- Broadcom NIC coredumps
- Broadcom sysfs files for RoCE statistics and tuning
- lldptool QoS configuration if installed
- niccli QoS configuration if installed

Installing the SOS Reporting Tool Automatically

The `bcm_sosreport` tool is prepackaged as a RPM and DEB package within the `utils/bcm_sosreport` folder of the software release archive. Install this package using the following command:

```
RPM: yum install $REL_DIR/utils/bcm_sosreport/bcm_sosreport-<version>.noarch.rpm
DEB: apt-get install $REL_DIR/utils/bcm_sosreport/bcm_sosreport-<version>.deb
```

Installing the SOS Reporting Tool Manually

To install the SOS Reporting Tool manually, see the following sections:

Installing the sosreport from pip

The `bcm_sosreport` package is dependent on the sosreport tool. If the system does not have an sosreportpackage (such as SUSE), manually install sosreport and bcm_sosreport.

1. Install the pip utility package (python3-pip) per the project documentation located here.
2. Install sosreport using pip: `pip3 install sosreport`

Installing the bcm_sosreport Package

After the sosreport package is installed with pip, the `bcm_sosreport` package can be installed with yum or apt-get as follows:

```
RPM: yum install /utils/bcm_sosreport/bcm_sosreport-<version>.noarch.rpm
DEB: apt-get install/utils/bcm_sosreport/bcm_sosreport-<version>.deb
```

Installing bcm_sosreport Manually

If it is not possible to install the `bcm_sosreport` .deb or .rpm package, manually extract the tar file and install each file as follows:

```
tar xf bcm_sosreport-<VERSION>.tar.gz
cd bcm_sosreport-<VERSION>
cp etc/sos/extras.d/*/etc/sos/extras.d/
cp opt/bcm_sosreport/* /opt/bcm_sosreport/
cp usr/bin/bcm_sosreport /usr/bin/
chmod +x /usr/bin/bcm_sosreport
```

Running the Report

After `bcm_sosreport` is installed, run a report using the following command:

```
bcm_sosreport
```

To view additional options, use the following command:

```
bcm_sosreport -h
```

Using the Report

The `bcm_sosreport` collects system data and then generates a .tar.gz file. The name of this file is shown by `bcm_sosreport` at the end of the collection. If working with a Broadcom support engineer, make this file available to them.

Viewing the File Contents

To view the contents of the file, copy it to a convenient system and then extract it with the following command:

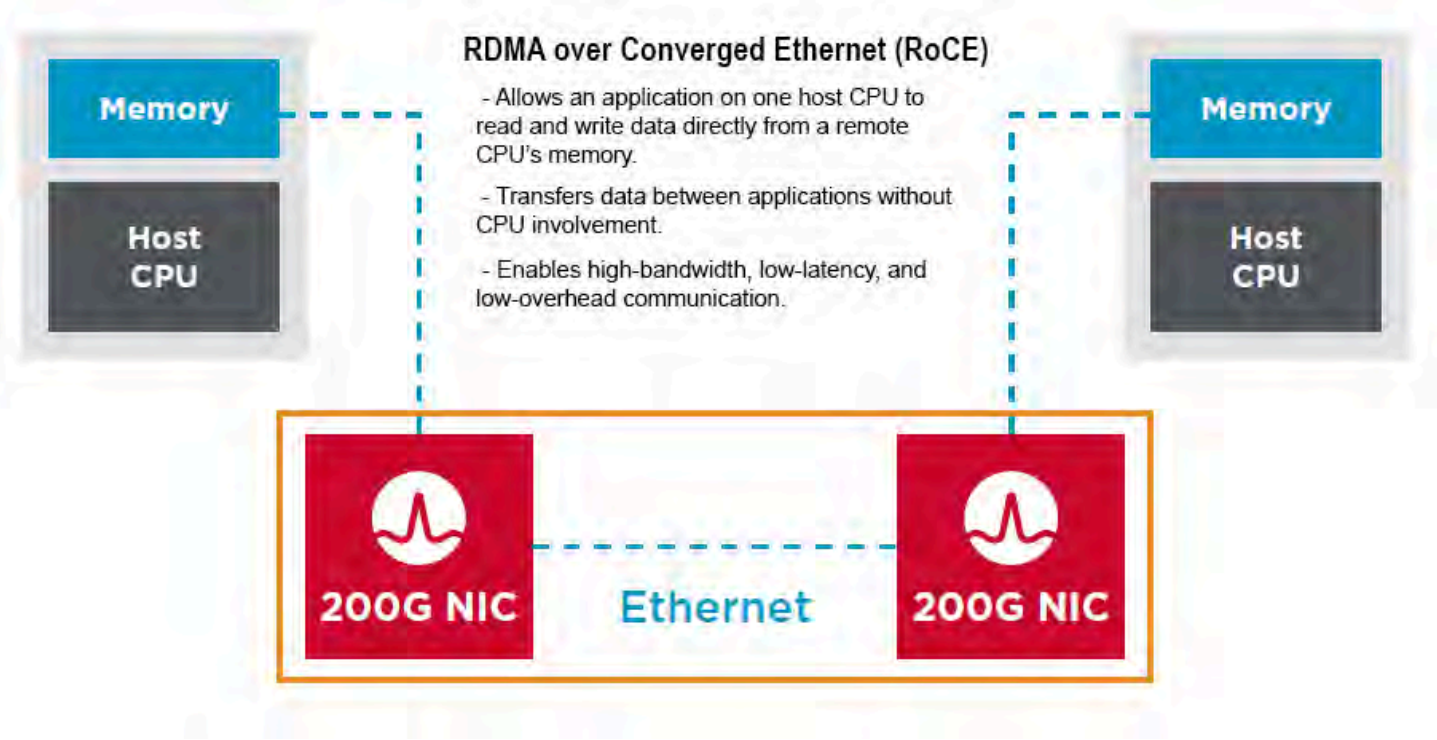
```
tar xf <FILE>
```

A directory is created with the same name as the file (minus .tar.gz) with a partial copy of the target server's file system, plus additional log files, and collected device logs in the "tmp" directory.

RDMA over Converged Ethernet (RoCE)

Provides information on how to set up and use Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) with Broadcom network adapters.

RoCE (RDMA over converged Ethernet) is a complete hardware offload feature supported on Broadcom Ethernet network adapters, which allows RDMA functionality over an Ethernet network. RoCE helps to reduce CPU workload as it provides direct memory access for applications bypassing the CPU. As the packet processing and memory access are done in hardware, RoCE allows for higher throughput, lower latency, and lower CPU utilization on both the sender and the receiver side, which are critical for Machine Learning (ML/AI), Storage, and High Performance Compute (HPC) applications.



Broadcom's Ethernet network adapters support complete hardware offload for RDMA. Broadcom NICs supporting RDMA are referred to as RNICs.

RoCE functionality is available for both user mode and kernel mode applications. RoCE is supported under Linux, Windows, and VMware operating systems.

Why use RDMA?

- Allows an application on one host CPU to read and write data directly from a remote CPU's memory.
- Transfers data between applications without CPU involvement.
- Enables high bandwidth, low latency, and low overhead communication.

See the following sections for RoCE requirements, installation, and configuration information:

- RDMA over Converged Ethernet Feature in Ethernet Network Adapters

Requirements

- Supported Devices
- Supported RoCE Protocols
- Supported Operating Systems
- Hardware Requirements

Software Installation

- Installing the RoCE Software
  - Linux Driver
  - VMware Driver
  - Windows Driver

Network Switch Configuration

- Configuring the Network Switch

Ethernet Network Adapter Configuration

- Configuring RDMA over Converged Ethernet (RoCE)
- Performance Profiles
- Tuning RoCE for Ethernet Network Adapters

RoCE Statistics

- Gathering RoCE Statistics
- RoCE Statistics Definitions

Other RoCE Topics

- RoCE Specific FAQs
- Example RoCE + TCP Configuration on Ethernet Network Adapters
- Configuring Peer Memory Direct with BCM95750X/BCM957608 Network Adapters

Configuring RoCE on Linux

Provides instructions for configuring RoCE on Linux.

As described in Installing the L2 and RoCE Drivers Automatically, the installer can be used to install RoCE with the following default parameters:

- RoCEv2 (RDMA over IPv4) enabled
- Congestion Control and PFC enabled
- RoCE traffic tagged with DSCP value 26 on priority 3
- RoCE CNP traffic tagged with DSCP value 48 on priority 7
- 1500-byte MTU

In addition, the automated installer has options that allow for non-default configuration settings as described in the following table.

Table 59: Common Changes from the Default Configuration using the Automated Installer

Control Type	Behavior	Example Application
Disable PFC (Congestion Control Only)	Removes risk of traffic blocking and congestion spreading. Incompatible with unreliable RDMA connection types. (UD, UC)	Specify `-o` ECN to installer
Disable ECN (PFC Only)	Improves throughput for very bursty traffic. Incompatible with multihop switch fabrics.	Specify `-o` PFC to installer
Use VLAN tags instead of DSCP	Appends VLAN header to RoCE traffic. Uses VLAN priority field to classify packets instead of IP header DS field.	Specify `-q <VID>` to installer
Use Jumbo Frames	Increases maximum throughput. All devices on the network must use the same MTU. Typical large value is 9000.	Specify `-m <MTU>` to the installer

The installer can be safely re-run at any time to change the RoCE configuration based on the parameters the installer uses.

Note: Ensure that the RNIC configuration matches the switch fabric configuration.

Broadcom also provides tools to modify default QoS parameters on the RNIC to match the network configuration. The parameters that can be changed are as follows:

- Congestion Control: Enable/Disable
- Priority Flow Control: Enable/Disable
- RoCE and General IP traffic priority
- Bandwidth allocation ratio for RoCE and IP using ETS
- DSCP priority for RoCE and CNP traffic
- VLAN priority for RoCE and CNP traffic
- Congestion control internal tunings

The tools provided for the previous parameters are as follows:

- `niccli` – This tool is used to configure and query QoS mappings, configure PFC, and configure ETS on the RNIC
- `lldpagent` – `niccli` is preferred over lldpagent
- `bnxt_setupcc.sh` – This script is provided for ease of use to modify RoCE QOS and Congestion Control-related settings and abstracts `niccli` or `lldpagent` usage.

Configuring the ETS Manually

After loading the RoCE driver, configure the ETS manually. Use existing tools like NICCLI/dcb for programming or install a config package provided by Broadcom.

1. The `bnxt_re_conf` package installs a udev rule which operates when a RoCE device is added to the system. The rule internally calls `bnxt_setupcc.sh` and does the ETS programming. The default values are read from a conf file(`/etc/bnxt_re/bnxt_re.conf`).
2. The bnxt_setupcc.sh can be run as follows (update the parameters accordingly):

```
sh bnxt_setupcc.sh -d rocep10s0f0 -i ens2f0np0 -r 3 -s 26 -c 7 -p 48 -u 4 -C 1
```

Modifying RoCE NVM Configuration Using NICCLI

Provides information on modifying the RoCE NVM configuration using NICCLI.

The `niccli` tool is used to modify the RNIC FW NVM configuration and also upgrade the firmware on the RNIC. The firmware NVM CFGs related to RoCE are shown in the following table.

Table 60: NVRAM Firmware Parameters

NVM Parameter Name	NVM Parameter Description	NVM Parameter Supported Values	Configured Value
`support_rdma`	This option indicates if an interface/PF supports RDMA.	Disabled (0) Enabled (1)	1 for BCM5750X, 0 for BCM574XX
`dcbx_mode`	This option specifies the DCBX mode configuration per port.	Disabled (0) Enabled (IEEE only) (1) CEE (only) (2) Both (IEEE preferred with fallback to CEE) (3)	0
`lldp_nearest_bridge`	This option allows enabling Link Layer Data Protocol (LLDP) on the nearest bridge, per port.	Disabled(0) Enabled (IEEE only)(1) CEE (only)(2) Both (IEEE preferred with fallback to CEE)(3)	0
`lldp_nearest_non_tpmr_bridge`	This option allows enabling Link Layer Data Protocol (LLDP) on non-TPMR (two port MAC Relay) bridge, per port.	Disabled(0) Enabled(1)	0
`disable_rdma_sriov`	This option controls if RDMA SRIOV is enabled on an interface.	Disabled(0) Enabled(1)	1

The option `support_rdma` must be enabled for an interface for RoCE to be enabled on the interface. With the use of the automated Linux installer on an interface, the `support_rdma` option is automatically set to 1. With manual installation, the option can be enabled as follows:

NICCLI Command

```
niccli -i <index> nvm --setoption support_rdma --scope 0 --value 1
```

After the NVM CFG changes have been made, the tool prompts for a host restart for the new NVM CFG to take effect.

1. Enable RDMA support for PF0 using the following command: **NICCLI Command**

   ```
   niccli -i 1 nvm --setoption support_rdma --scope 0 --value 1
   niccli -i 1 fw --reset
   ```

   Note: The notification to reboot can be ignored if this is the only option being changed.
2. Disable in firmware DCBx with the following command:
   NICCLI Command

   ```
   niccli -i 1 nvm --setoption dcbx_mode --scope 0 --value 0
   ```

   Note: This procedure must run after the bnxt_en driver has loaded.
3. Reboot the system to apply the changes.

Modifying QoS Configuration Using NICCLI

Provides information on modifying QoS configuration using the NICCLI utility.

NICCLI can be used for QoS configuration, setting QoS mappings, priority flow control, and configuring ETS. QoS configures the settings on the NIC and requires a symmetrical configuration on network switches. NICCLI can configure the APPTLVs, PC, and ETS. The generic syntax for using NICCLI is as follows:

Modifying QoS Configuration Using NICCLI
NICCLI can be used for QoS configuration, setting QoS mappings, priority flow control, and configuring ETS. QoS configures the settings on the NIC and requires a symmetrical configuration on network switches. NICCLI can configure the APPTLVs, PC, and ETS. The generic syntax for using NICCLI is as follows:

NICCLI Command

```
niccli -i <index> <command> [-options [....]]
```

Commonly used commands are described in the following table.

Table 61: NICCLI Commands

Command	Description
version	This command displays the program version details.
`qos --ets`	This command configures the priority to TC and bandwidths. This command configures only TSA and Bandwidths **Syntax**: `niccli -i <index> --ets --tsa <tc[0-7]:[ets|strict], ...> --up2tc <priority[0-7]:tc>, ...> --tcbw <list>` with comma separated and in units of percentage (%) **Example**: `niccli -i 1 qos --ets --tsa 0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict --up2tc 0:0,1:0,2:0,3:0,4:0,5:1,6:0,7:0 --tcbw 70,30`
`qos --pfc`	This command enables PFC on given priority. Valid values are 0 to 7. **Syntax**: `niccli -i <index> qos --pfc --enable <0-7>` The values should be with comma separated **Example**: `niccli -i 1 qos --pfc --enable 5,6`
`qos --apptlv`	This parameter configures the priority of the APPTLV. **Syntax**: `niccli -i <index> qos --apptlv --add --app priority,selector,protocol>` **Example**: `niccli -i 1 qos --apptlv --add --app 3,1,35093 niccli -i 1 qos --apptlv --del --app 3,1,35093`
`qos --ets --show`	This command gets the configured priorities and bandwidth parameters.
`qos --tc --set --rate_limit`	This command sets the rate limit for each TC in units of percentage (%). **Example**: `niccli -i 1 ratelimit 80, 60, 70`
`qos --listmap --pri2cos`	This command dumps the supported dump strings. Supported dump string is pri2cos. **Syntax**: `niccli -i <index> qos --listmap --pri2cos` **Example**: `niccli -i 1 qos --listmap --pri2cos`

The `niccli` tool is packaged with Broadcom software/firmware and contains a README file that describes how to use the tool as well. For ease of use, the `bntxsetupcc.sh` shell script is provided that simplifies configuring these parameters.

Modifying RoCE Configuration Using bnxtsetupcc.sh

Provides information on modifying the RoCE configuration using the bnxtsetupcc.sh utility.

The `bnxt_setupcc.sh` tool uses either `niccli` or `lldptool` to configure network parameters with a single command. The `bnxtsetupcc.sh` script parameters are shown in the following table.

Table 62: bnxt_setupcc.sh Parameters

Option	Description
-C	VALUE Set CNP Service Type
-d	RoCE Device Name (for example, bnxt_re0, bnxt_re_bond).
-i	Ethernet Interface Name (for example, p1p1 or for bond, specify secondary interfaces like -i p6p1 -i p6p2)
-r [0-7]	RoCE packet priority
-s value	RoCE packet DSCP value
-c [0-7]	RoCE CNP packet priority
-p value	RoCE CNP packet DSCP value
-v 1	1 - Enable priority VLAN
-b value	RoCE Bandwidth percentage for ETS configuration. Default is 80%
-h	Display help
-m [1-3]	1. PFC only 2. CC only 3. PFC + CC mode
-u [1-4]	1. Use bnxtqos utility. Disables the lldptool if enabled (default). 2. Use lldptool. 3. Use Broadcom NICCLI utility. Disable lldptool if enabled. 4. Use dcb command -V Display version

The following example configures the specified interface to the default values of congestion control.

```
bnxt_setupcc.sh -d bnxt_re0 -i ens4f0np0 -m 3 -s 26 -p 48 -r 3 -c 7
```

The following example configures the specified interface to only use PFC (no CC).

```
bnxt_setupcc.sh -d bnxt_re0 -i ens4f0np0 -m 1 -s 26 -p 48 -r 3 -c 7
```

Note: The previous examples use `bnxt_re0` as the RoCE device name and `ens4f0np0` as the corresponding L2 interface.

DSCP and VLAN

Provides information on methods used for setting traffic priority for either VLAN or DSCP.

Note: VLAN-based PFC is not supported on BCM957608 devices.

Ethernet and Broadcom RNICs support two methods for marking traffic with a priority: VLAN and DSCP. If your network uses VLAN tagging for traffic, VLAN priority can be used to separate RoCE traffic from other traffic for congestion control. If your network does not use VLAN tags, DSCP must be used. In either case, the corresponding switch must be configured correctly to ensure the priority (VLAN or DSCP) maps to a specific traffic class priority on the switch that has ECN and/or PFC enabled and configured.

For RoCE deployment, specific VLAN PRI or DSCP values must be configured in the network and the host for RoCE, IP, and, RoCE CNP traffic. No industry standard defines specific VLAN PRI and DSCP values for RoCE traffic. To simplify RoCE deployment, the following recommended values of VLAN PRI and DSCP should be used (see the following table).

Note: When VLAN priority is used, take care when the frames are transmitted over the trunk network. When multiple trunks are not configured for correct priority, VLAN priority can be dropped.

Table 63: VLAN PRI and DSCP Recommended Values

Traffic	VLAN PRI	DSCP
RoCE (lossless)	3 (Flash)	26 (Flash)
IP (lossy)	0 (Best Effort)	0 (Best Effort)
RoCE CNP (lossy SP)	7 (Network Control)	48 (Network Control)

Bandwidth Allocation

Provides information on setting the minimum bandwidth using enhanced transmission selection (ETS).

Enhanced Transmission Selection (ETS) provides minimum bandwidth allocation values for both RoCE and IP traffic. ETS only comes into effect when both IP and RoCE traffic are scheduled for transmission, causing congestion in the server NIC. By default, 50% of traffic is allocated to RoCE, and 50% for IP.

Even if the use case is entirely RoCE, some bandwidth must remain allocated for IP for control of at least SSH and the RDMA connection manager.

The ETS ratio is set using either NICCLI or lldptool.

Advanced RDMA over Converged Ethernet (RoCE) Network Configuration

Provides information on advanced RoCE network configuration.

The default RoCE settings are usually close to optimal for most users and use cases. However, some users might wish to use different configurations to match existing network settings or optimize for specific applications. This section provides background on the internal Congestion Control and QoS implementation of Broadcom RNICs, and steps for changing each of the relevant settings.

The RoCE configuration defaults for the Linux drivers are:

- RoCEv2 (RDMA over IPv4) enabled
- Congestion Control and PFC enabled
- RoCE traffic tagged with DSCP value 26 on priority 3
- RoCE CNP traffic tagged with DSCP value 48 on priority 7
- 1500-byte MTU

Table 64: Common Changes from the Default Configuration

Control Type	Behavior	Example Applications
Disable PFC (Congestion Control Only)	Removes risk of traffic blocking and congestion spreading. Incompatible with unreliable RDMA connection types. (UD, UC)	Specify `-o ECN` to installer
Disable ECN (PFC Only)	Improves throughput for very bursty traffic. Incompatible with multihop switch fabrics.	Specify `-o PFC` to installer

Control Type	Behavior	Example Applications
Use VLAN tags instead of DSCP	Appends VLAN header to RoCE traffic. Uses VLAN priority field to classify packets instead of IP header DS field	Specify `-q <VID>` to installer
Use Jumbo Frames	Increases maximum throughput. All devices on network must use the same MTU. Typical large value is 9000.	Specify `-m <MTU>` to the installer

Ensure that the RNIC configuration matches the switch fabric configuration.

RoCE Congestion Control

Provides information on configuring RoCE congestion control.

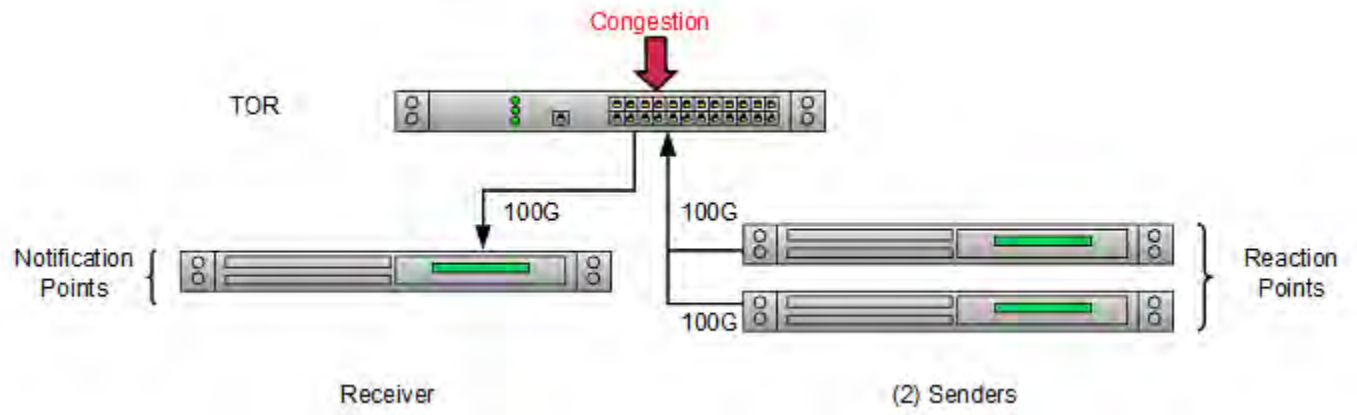
RoCE Congestion Control (CC) relies on ECN marking on the switch. When the switch accumulates excess incoming traffic in its buffers due to congestion, the switch output queue level rises. When the output queue level exceeds the configured minimum threshold, the switch marks the outgoing packets with ECN to indicate congestion. When the receiving NIC receives an ECN-marked packet it transmits a Congestion Notification Packet (CNP) to the sender, and specifically to the flow (QPN – QP number) to which the packet belongs. The sender is the NIC that transmitted the original marked packet. Upon receiving a CNP, the sending NIC reduces the transmission rate for the particular flow (QP).

The sender only reduces the rate on a particular flow (QP), to the one that the CNP was destined for. This behavior is critical to maintaining good network utilization by only penalizing (reducing rate) flows that contribute to congestion.

The following three network components are involved in avoiding congestion:

- The sending NIC – Reaction point as it reacts to congestion indication.
- The switch – Congestion point, where congestion is detected.
- The receiving NIC – Notification point simply transmits CNP when receiving ECN marked data packet.

Figure 34: Congestion Control Topology



The two main components that determine the congestion control efficiency are the marking policy at the switch and the sending NIC behavior. The set of rules for the behavior of the sending NIC when there is congestion and when there is not, is the congestion control algorithm.

The marking policy and the CC algorithm significantly affect the following performance factors:

- Network utilization.
- How likely switch buffers are to fill up.
- End-to-end latency through the network.

Each flow should react correctly and quickly to congestion, and quickly capture the available link bandwidth when no congestion exists, while keeping enough traffic flowing through the switch to use the link toward the receiving NIC, but with little queuing delay through the switch. Queuing delay through the switch is the result of the steady-state queue level, or how many packets are buffered in the switch. Minimizing the queue level during congestion directly reduces the end-to-end latency.

Deterministic Marking (DCQCN-D)

Provides information on the use of deterministic marking to reduce congestion.

In deterministic marking, after the queue level exceeds a configured level, all packets are marked. Therefore, all flows receive immediate congestion indication and reduce their rate. The BCM95741X Ethernet network adapter adopts a CC algorithm designed for deterministic marking policy and very low queuing latency called DCQCN-D. The BCM95750X/BCM957608 supports a CC algorithm designed for deterministic marking policy and very low queuing latency called DCQCN-D. With this algorithm, the marking threshold is very low (a few tens of KB), and therefore, flows react quickly to congestion while keeping the number of packets buffered in the queue very small. With DCQCN-D, queuing latency through congested switches is very low. This approach can include a cost of slightly lower egress link utilization.

The BCM95750X/BCM95741X Ethernet network adapter provides flexibility in CC settings to control how aggressively flows start transmission versus how likely the switch buffers might fill up in the event of a large incast. When a large number of flows on the network start transmission toward the same destination at the same time, the switch needs to absorb the initial, or transient, excess traffic until the flows receive congestion indication and reduce rate. The more aggressive (higher) initial rate, the more buffering is needed for a given number of starting flows. If switch buffering is exhausted, the switch asserts flow control if it is configured to do so, or drops packets. Depending on the expected burstiness of traffic on the network, the optimal setting can be selected.

Probabilistic Marking (DCQCN-P)

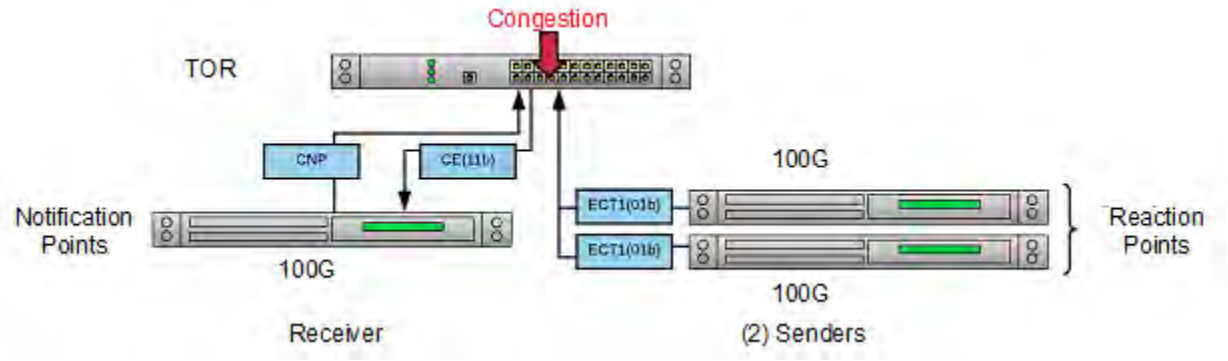
Provides information on the use of probabilistic marking to reduce congestion.

In probabilistic marking, minimum and maximum marking thresholds are specified at the switch together with maximum marking probability. When the switch queue crosses the minimum threshold, marking starts at low probability which linearly increases between 0 and max probability in the range between min. and max. marking thresholds. Once the maximum threshold is reached, every packet is marked (for example, 100% marking). With this algorithm, the minimum marking threshold is higher than the fixed threshold of DCQCN-D (a few hundred kB). The BCM95750X/BCM957608 Ethernet network adapter is configured to DCQCN-P and reacts to congestion notifications accordingly. With DCQCN-P, the steady state queue level is higher, thus end-to-end latency experienced by other applications is higher and could lead to more interference between applications when they run concurrently on the network. Egress link utilization can be slightly higher and flows at a reduced rate only with sustained congestion when the switch queue level reaches the minimum marking threshold.

Traffic Control Synopsis

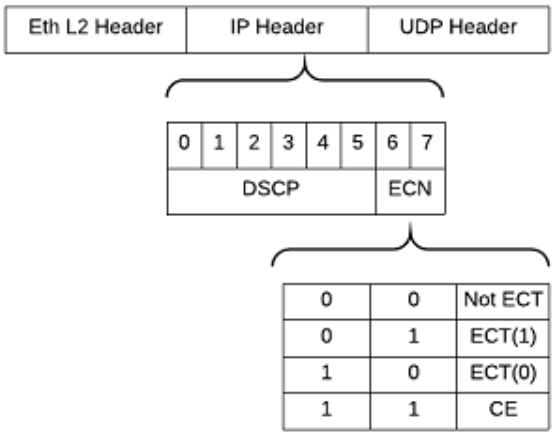
An example scenario of congestion is provided to demonstrate the behavior and function of congestion control.

Figure 35: Congestion Control Example Network



1. Three servers connect to a switch and are configured for RoCE with the default settings (CC and PFC are enabled). The switch is configured to enable ECN and PFC on the RoCE priority.
2. Network congestion is caused by two servers transmitting data to the third server simultaneously, at a speed above the link rate of the receiver. This scenario causes packets to be buffered for a time in the switch before being transmitted. If the buffer fills up, it drops packets on ingress, which causes a cycle of retransmits and additional congestion.
3. If ECN is enabled (default), the initial QP transmit rate begins at a fraction of line rate, and quickly ramps up until congestion is encountered or line rate is reached. If ECN is not enabled, RDMA transmissions are sent at line rate.
4. RoCE packets from the transmitting hosts are tagged as ECN enabled by setting the ECN field in the IP header to ECT1(01b) or ECT0(10b).

   Figure 36: ECN Congestion Management



5. As the egress buffer of the switch reaches the ECN congestion marking threshold as defined by the user, the switch starts marking the ECN bits to CE(11b) to inform the notification point server (receiver) that congestion has been detected for that port.
6. If the switch buffer reaches the PFC threshold, PFC frames are sent to the transmitting servers, instructing them to stop transmitting. This process prevents the switch from dropping packets, but can lead to increased latency and latency jitter. Switch ingress ports that are not contributing significantly to congestion can also transmit PFC frames to their link partners. The switch cannot predict the destination of packets that it receives, so the switch must make sure that arriving packets can land in switch memory. This behavior can result in head-of-line blocking, because packets destined for noncongested destinations might be held at the link partner.

7. If the network is multiswitch, congestion spreading can occur because of throttled switch links from congestion on only one switch egress port.
8. Upon the reception of a RoCE packet with the ECN CE flag set, the notification point (receiving server) transmits CNP packet(s) to the reaction points (transmitting servers) to signify congestion.
9. The reaction point correlates received CNP packets to a specific QP, and adjusts the transmit rate of that QP to reduce congestion based on the algorithm described in Deterministic Marking (DCQCN-D).
10. QP transmit rate is frequently re-assessed to allow increased speed during times of reduced congestion.

Congestion Control Tuning Parameters

Provides a list of configurable congestion control parameters. Updating the parameters is accomplished through configfs files.

Note: It is not normally necessary to adjust these values. These values should only be modified when requested by the Broadcom Support team.

Table 65: Congestion Control Tuning Parameters

CC Parameter	Description
`cc_mode`	0 for **DCQCN-D** algorithm (with deterministic marking), 1 for **interop with DCQCN-P** (probabilistic marking). Default 1.
`ecn_enable`	Enable congestion control.
`g`	Running average weight in computing congestion probability (cp). The weight is 2^g, g between 1 and 9, default 8.
`init_cr`	Current rate after QP creation and after QP has not transmitted for more than `inctivity_th`. Value between 0 and 1023 as fraction of full link BW. The driver configures the value when the PF is enabled.
`init_tr`	Target rate after QP creation and after QP has not transmitted for more than `inctivity_th`. Value between 0 and 1023 as fraction of full link BW.
`rtt`	Time period (µs) over which cnp and transmitted packets counts accumulate. At the end of Rtt, the ratio between CNPs and TxPkts is computed and the CP is updated, default 40 µs.
`inact_cp`	Inactivity time after which the CC parameters of a QP are re-initialized, default 10000 µs.
`cnp_dscp`	DSCP value for RoCE congestion notification packets.
`roce_dscp`	DSCP value for RoCE packets.
`cnp_prio`	Priority for RoCE congestion notification packets.
`roce_prio`	Priority for RoCE packets.
`apply`	Applies the settings.

Changing Congestion Control Mode Settings

Provides information on changing the congestion control mode setting.

`Cc_mode` defaults to the DCQCN-P (1) algorithm for the BCM575XX. By setting `cc_mode` to DCQCN-D (0, previously known as DCTCP) algorithm, five other advanced parameters are automatically set to optimal values matching the new `cc_mode`. The following are the advanced parameters and their values for `cc_mode`.

Table 66: Advanced Parameters for cc_mode

Advanced Parameters	DCQCN-P Mode	DCQCN-D Mode
cp bias state	Disabled	Enabled
target rate update mode	1	0
log of cp to cr ratio	7	1
current rate level to max cp	0x3ff	0x280
reduce to init rtts threshold	0x3eb	0x15e

The following commands set the `cc_mode` to DCQCN-P algorithms:

```
mkdir -p /sys/kernel/config/bnxt_re/bnxt_re0
cd /sys/kernel/config/bnxt_re/bnxt_re0/ports/1/cc/
echo -n 1 > cc_mode
echo -n 1 > apply
```

The following commands set the `cc_mode` to DCQCN-D algorithms:

```
mkdir -p /sys/kernel/config/bnxt_re/bnxt_re0
cd /sys/kernel/config/bnxt_re/bnxt_re0/ports/1/cc/
echo -n 0 > cc_mode
echo -n 1 > apply
```

The following command shows the value of those parameters:

```
mkdir -p /sys/kernel/config/bnxt_re/bnxt_re0
cd /sys/kernel/config/bnxt_re/bnxt_re0/ports/1/cc/
echo -n 1 > advanced
echo -n 1 > apply
cat apply
```

Sample Output for DCQCN-P Mode

`Cc_mode` and five advanced parameters **bolded** are shown in the following example output:

ecn status	: Enabled
ecn marking	: ECT(1)
congestion control mode	**: Probabilistic**
send priority vlan (VLAN 0)	: Disabled
running avg. weight(g)	: 8
inactivity threshold	: 10000 µs
initial current rate	: 0xc8
initial target rate	: 0x320
round trip time	: 45 µs
cnp header ecn status	: ECT(1)
rtt jitter	: Enabled
link bytes per µs	: 0x30d4 byte/µs
current rate width	: 0xe bits
minimum quota period	: 0x4

maximum quota period	: 0x7
absolute maximum quota period	: 0xff
64B transmitted in one rtt	: 0x3d40
extended inactivity threshold	: 0x0
minimum time between cnps	: 0x0 µs
initial congestion probability	: 0x3ff
target rate update mode	**: 1**
target rate update cycle	: 0x0
fast recovery rtt	: 0x5 rtts
active increase time quanta	: 0x1
reduc. relax rtt threshold	: 0x2 rtts
additional relax cr rtt	: 0x50 rtts
minimum current rate threshold	: 0x0
bandwidth weight	: 0x5
actual current rate factor	: 0x0
current rate level to max cp	**: 0x3ff**
cp bias state	**: Disabled**
log of cr fraction added to cp	: 0x3
cr threshold to reset cc	: 0x32a
target rate lower bound	: 0x1
current rate probability factor	: 0x3
target rate probability factor	: 0x5
current rate fairness threshold	: 0x64
reduction divider	: 0x1
rate reduction threshold	: 0x0 cnps
extended no congestion rtts	: 0x8 rtt
log of cp to cr ratio	**: 0x7**
use lower rate table entries	: Disabled
rtts to start cp track cr	: 0x1a4 rtt
first threshold to rise ai	: 0x40 rtt
second threshold to rise ai	: 0x80 rtt
actual rate base reduction threshold	: 0x15e rtt
first severe cong. cr threshold	: 0x0
second severe cong. cr threshold	: 0x0
cc ack bytes	: 0x44
reduce to init rtts threshold	**: 0x3eb rtt**
roce prio	: 3
roce dscp	: 26
cnp prio	: 7
cnp dscp	: 48

When the link speed, link status, or PFC enable state changes on a port, the `cc_mode` and the five advanced parameters are automatically set to the default values for DCQCN-P algorithms.

Setting Congestion Control Parameters

Provides information on manually setting congestion control parameters writing to the device's configfs files for BCM95741X devices.

The following commands set the default values:

```
mkdir -p /sys/kernel/config/rdma_cm/bnxt_re0
echo RoCE v2 > /sys/kernel/config/rdma_cm/bnxt_re0/ports/1/default_roce_mode

mkdir -p /sys/kernel/config/bnxt_re/bnxt_re0
cd /sys/kernel/config/bnxt_re/bnxt_re0/ports/1/cc/

echo -n 0x1a > roce_dscp
echo -n 0x3 > roce_prio
echo -n 0x1 > disable_prio_vlan_tx
echo -n 0x1 > ecn_enable
echo -n 0x30 > cnp_dscp
echo -n 0x7 > cnp_prio

#After setting all the above parameters, apply the values to HW
echo -n 0x1 > apply
```

Quality of Service

Provides information on Quality of Service (QoS) policies.

Broadcom RNICs support multiple Class of Service Queues (CoSQ) per port. QoS policy is based on TCs that are mapped 1:1 to CoS queues. VLAN priority to CoSQ mapping is supported as well as DSCP to CoSQ mapping. At the NIC driver level, each transmit ring or RoCE Queue Pair (QP) is associated with a particular CoSQ. The NIC driver performs this configuration. On each port, the NIC schedules traffic across multiple CoSQs.

A finite amount of on-chip buffering is available for transmitting and receiving packets. These buffers are referred to as TX and RX MBUFs. On the transmit side, the TX MBUF stages packets that are transmitted on the network. The host software initiates the transmission of these packets. The RNIC processes, stages, and builds packets in the TX MBUF before they are transmitted on the network. On the receive side, the RX MBUF stages packets received from the network before the packets or packet payloads are transferred into the host memory.

A CoS profile (also known as MBUF profile) is a CoSQ and MBUF configuration to enforce a specific QoS policy on a port. An MBUF configuration profile has the following attributes:

- The number of CoSQs.
- The specific CoSQ enablement.
- The type of service such as lossy or lossless per CoSQ.
- The buffer configuration for each CoSQ.
- Pause/PFC threshold for each CoSQ.

BCM95741X Ethernet network adapters support three transmit and receive CoSQs for each Ethernet port: 0, 4, and 5. BCM95750X/BCM957608 Ethernet network adapters support eight transmit and receive CoSQs for each Ethernet port: 0 through 7. By default, all CoSQs are configured for weighted-fair-queueing (WFQ), with priority 0 traffic mapped to CoSQ 4. Configure priority-to-CoSQ mappings to differentiate traffic types among the allocated class of service queues.

When the RoCE `bnxt_re` driver is loaded, CoSQ 0 is configured for lossless traffic, and CoSQ 5 is changed from WFQ to strict priority (SP) for CNP processing.

RoCE and CNP traffic can be tagged with different DSCP values, or use VLAN tags instead.

Viewing/Changing the RoCE DSCP/VLAN Settings

To dump the congestion control PFC and ETS-related settings on the network adapter, use the `niccli` tool as follows:

```
niccli -i 1 qos --ets --show
IEEE 8021QAZ ETS Configuration TLV:
         PRIO_MAP: 0:0 1:0 2:0 3:1 4:0 5:0 6:0 7:2
         TC Bandwidth: 50% 50% 0% 0% 0% 0% 0% 0%
         TSA_MAP: 0:ets 1:ets 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
IEEE 8021QAZ PFC TLV:
         PFC enabled: 3
IEEE 8021QAZ APP TLV:
         APP#0:
          Priority: 7
          Sel: 5
          DSCP: 48

         APP#1:
          Priority: 3
          Sel: 5
          DSCP: 26

         APP#2:
          Priority: 3
          Sel: 3
          UDP or DCCP: 4791

TC Rate Limit: 100% 100% 100% 0% 0% 0% 0% 0%

 niccli -i 1 qos --dscp2prio

dscp2prio mapping:
          priority:7  dscp:48,
          priority:3  dscp:26

niccli -i 1 qos --listmap --pri2cos

Base Queue is 0 for port 0.
------------------------
Priority   TC   Queue ID
------------------------
 0         0    4
 1         0    4
 2         0    4
 3         1    0
 4         0    4
 5         0    4
 6         0    4
 7         2    5
```

Note: The previous `niccli` examples use `index 1` as an example. The actual index should be used.

Validating RDMA over Converged Ethernet (RoCE) Network on Linux

Provides information on validating the RoCE network on Linux.

With the switches, servers, and NICs configured, RDMA applications can be deployed to run over the RoCE network. Any libverbs-linked application can be used, with appropriate changes for using IP addresses rather than GUIDs for end-point addressing.

Ensure that the RoCEv2 GID is correct (`bnxt_re0` is used as an example of a RoCE port).

```
# cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/3
RoCEv2
```

The `ibv_devices` command is used to check the attributes of the available devices.

```
# ibv_devices
device node GUID
------ ----------------
bnxt_re0 6e92cffffe633c70d
```

Use `ibv_devinfo` to check the attributes of the devices:

```
# ibv_devinfo
hca_id: bnxt_re0
transport: InfiniBand (0)
fw_ver: 233.0.150.0
node_guid: 6e92:cfff:fe63:3c70
sys_image_guid: 6e92:cfff:fe63:3c70
vendor_id: 0x14e4
vendor_part_id: 5984
hw_ver: 0x11
phys_port_cnt: 1
port: 1
state: PORT_ACTIVE (4)
max_mtu: 4096 (5)
active_mtu: 1024 (3)
sm_lid: 0
port_lid: 0
port_lmc: 0x00
link_layer: Et
```

A typical application of exercising the RoCE interface would be to use `rping` or `ib_write_bw/etc` from perftest. Retrieve perftest from GitHub or through most Linux package repositories, such as:

```
Fedora/RHEL/CentOS: yum install perftest
Ubuntu: apt install perftest
```

Running the Server rping Traffic Test

To run the server `rping` traffic test, use the following commands:

```
Server: rping -s -d -v -a <ip of server bnxt interface>
```

Example:

```
rping -s -a 192.172.1.1 -Vv -C 3
server ping data: rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
server ping data: rdma-ping-1: BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
server ping data: rdma-ping-2: CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
```

Running the Client rping Traffic Test

To run the client `rping` traffic test, use the following commands:

```
Client: rping -c -d -v -a <ip of server bnxt interface>
```

Example:

```
rping -c -a 192.172.1.1 -C 3 -vV
ping data: rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
ping data: rdma-ping-1: BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
ping data: rdma-ping-2: CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
```

The perftest tools can be run as:

```
# ib_write_bw -d bnxt_re0 -x 3 -F --report_gbits -p 1800 -s <message_size> -q <num of qps> -D <duration>
ib_write_bw -d bnxt_re0 -x 3 -F --report_gbits -p 1800 -s 1048576 -q 16 (server)
ib_write_bw -d bnxt_re0 -x 3 -F --report_gbits -p 1800 -s 1048576 -q 16 192.168.1.68 (client)
```

Note: In the previous example, 192.168.1.68 is the IP address of the server.

For a complete understanding of the parameters used by `ib_write_bw/lat` and other IB utilities, refer to the Linux man pages.

Configuring RoCE on VMware

Provides instructions for configuring RoCE for VMware.

Refer to Vmware.com for additional information on setting up and using Paravirtualized RDMA (PVRDMA) network adapters.

Prerequisites

- RDMA must be enabled in the device NVRAM via BIOS setup.

Configuring a Virtual Center for PVRDMA

Use the following steps to configure a Virtual Center for PVRDMA:

1. Creating a distributed virtual switch (DVS) requires a DVS for PVRDMA.
2. Add the host to the DVS.

Tagging vmknic for PVRDMA on ESX Hosts

To tag a vmknic for PVRDMA to use on ESX hosts:

1. Select the host and right-click **Settings** to switch to the settings page of the **Manage** tabs.
2. In the **Settings** page, expand **System** and click **Advanced System Settings** to show the advanced system settings key-pair value and its summary.
3. Click **Edit** to bring up the **Edit Advanced System Settings**.
   Filter on **PVRDMA** to narrow all the settings to just Net.PVRDMAVmknic.
4. Set the Net.PVRDMAVmknic value to vmknic.

Setting the Firewall Rule for PVRDMA

Use the following steps to set the firewall rule for PVRDMA:

1. Select the host and right-click **Settings** to switch to the settings page of the **Manage** tabs.

2. In the **Settings** page, expand **System** and click **Security Profile** to show the firewall summary.
3. Click **Edit** to bring up the **Edit Security Profile**.
4. Scroll down to find pvrdma and check the box to set the firewall.

Adding a PVRDMA Device to the VM

Use the following steps to add a PVRDMA device to the VM:

1. Select the VM and right-click **Edit Settings**.
2. Add a new Network Adapter.
3. Select the network as a **Distributed Virtual Switch** and **Port Group**.
4. For the **Adapter Type**, select **PVRDMA** and click **OK**.

Configuring the VM on Linux Guest Operating System

Note: The user must install the appropriate development tools including git before proceeding with the following configuration steps.

1. Download the PVRDMA driver and library by using the following commands:
   ```
   git clone git://github.com/linux-rdma/rdma-core.git
   ```
2. Compile and install the PVRDMA guest driver and library.
3. To install the driver, execute `make && sudo insmod pvrdma.ko` in the driver directory. The driver must be loaded after the paired `vmxnet3`driver is loaded.
   Note: The installed RDMA kernel modules might not be compatible with the PVRDMA driver. In this case, remove the current installation and restart. Follow the installation instructions. Refer to the README in the driver's directory for more information about the different RDMA stacks.
4. To install the library, execute `./autogen.sh && ./configure --sysconfdir=/etc && make && sudo make install` in the directory of the library.
   Note: The installation path of the library needs to be in the shared library cache. Follow the instructions in the INSTALL file in the library's directory.
   Note: The firewall settings might need to be modified to allow RDMA traffic. Ensure the proper firewall settings are in place.
5. Add the `/usr/lib` in the `/etc/ld.so.conf` file and reload the `ldconf` by running `ldconfig .`
6. Load IB modules using `modprobe rdma_ucm .`
7. Load the PVRDMA kernel module using `insmod pvrdma.ko .`
8. Assign an IP address to the PVRDMA interface.
9. Verify whether the IB device is created by running the `ibv_devinfo -v` command.

PFC/ECN Configuration

This section provides information on PFC/ECN configuration. The following modes are supported:

- VLAN-PCP based PFC only
- DSCP-based PFC + CC mode
- DSCP-based CC only
- DSCP-based PFC only
- VLAN-PCP based PFC + CC mode - Currently not supported in ESX
- The BMC-ESX bnxtrocecli vib module is required for checking and configuring the DSCP-based PFC + CC mode. Ensure that this module is installed.

Common Settings

Configure the following common settings:

1. Enable the firmware-based DCBx for all modes listed using the following command: `#esxcfg-module -s "disable_dcb=0 enable_host_dcbd=0" bnxtnet .`
2. Crosscheck the DCBx settings using the following commands:
   ```
   #esxcli network  nic dcb status get -n vmnic2
   #esxcli bnxtnet dcb get -n vmnic2
   ```
3. In case the switch does not support RoCE-App-TLV, specify the same PRI on module-param using the following command: `#esxcfg-module -s "roce_pri=3" bnxtnet`
4. Enable RoCE-App-TLV and ECN on the external switch ports for all modes listed in the following procedure.

VLAN-PCP Based PFC Only

To configure VLAN-PCP based PFC only:

1. Do not enable DSCP.
   ```
   #esxcfg-module  -s "dscp_trust=-1" bnxtroce
   ```
   or
   ```
   #esxcfg-module  -s "" bnxtroce
   ```
2. Enable PFC on the switch.

DSCP-Based PFC + CC Mode

To configure DSCP-based PFC + CC mode:

1. Configure DSCP with a desired value using the following command:
   ```
   esxcfg-module  -s "dscp_trust=48" bnxtroce
   ```
2. Check and configure the DSCP-to-PRI mapping and DSCP of the CNP using key-val interface as shown in the following commands:
   ```
   /usr/lib/vmware/vmkmgmt_keyval/vmkmgmt_keyval -i "key_val_vmrdma0/broadcom" -k dscp_to_pri_map -g
   /usr/lib/vmware/vmkmgmt_keyval/vmkmgmt_keyval -i "key_val_vmrdma0/broadcom" -k dscp_to_pri_map -s "25 5"
   ```
3. Check and configure the CNP settings using the following commands:
   ```
   #esxcli bnxtroce cnp get -n vmrdma0
   #esxcli bnxtroce cnp  set -p 3 -C 1 -P 34 -E 0x1 -n vmrdma0
   ```

DSCP-Based CC Only

The configuration is the same as shown in the previous section, but PFC is disabled on the external switch ports.

DSCP-Based PFC Only

The configuration is the same as shown in the previous section, but disable CC/ECN on the external switch ports.

Configuring RoCE on Windows

This section provides information on configuring RoCE in Windows for Broadcom Ethernet network adapters.

Prerequisites

- RDMA must be enabled in the device NVRAM via BIOS setup.

Enabling RDMA on the System and Ethernet Network Adapter

There are three modes of RDMA support in Windows. All three require the following steps to enable RDMA on the system and the adapter.

1. From the **Advanced Properties** page of the Broadcom Ethernet network adapter, confirm and enable the following options:

- Network Direct Functionality – TRUE
- Network Direct Technology – RoCEv2 (RoCEv1 is not supported)

These steps can also use PowerShell commands:

- Show all net adapters on the system:
  ```
  Get-NetAdapter
  ```
- Enable RDMA on the required NIC:
  ```
  Enable-NetAdapterRdma -Name "SLOT 5 Port 1"
  ```
- Ensure RDMA is enabled on the NIC:
  ```
  Get-NetAdapterRdma
  ```
  Note: The output is enabled=TRUE.
- Verify that SMBD listeners are present:
  ```
  netstat -xan
  ```

Enabling RDMA in the NVRAM via NICCLI

RDMA must be enabled in the device NVRAM. Three settings in the NVRAM must be enabled: A global on/off setting that affects all functions, a per function on/off setting, and an on/off setting for RDMA in VFs.

1. Determine the MAC and its respective index for each function by using the following command:
   ```
   niccli --list_devices or niccli --list
   ```
2. Use the following command to determine if the device is RDMA capable. This is the global setting. If RDMA capability is turned off, it will be off for all functions regardless of the function's RDMA setting.
   ```
   niccli -i <index> nvm --getoption rdma_capable
   ```
3. To check the per-function RDMA setting, use the following commands:
   a. Query PCI function 0
      ```
      niccli -i <index> nvm --getoption support_rdma --scope 0
      ```
   b. Query PCI function 1
      ```
      niccli -i <index> nvm --getoption support_rdma --scope 1
      ```
4. To enable or disable the per-function RDMA setting, use the following commands:
   a. Turn on RDMA for PCI function 0
      ```
      niccli -i <index> nvm --setoption support_rdma --scope 0 --value 1
      ```
   b. Turn on RDMA for PCI function 1
      ```
      niccli -i <index> nvm --setoption support_rdma --scope 1 --value 1
      ```
5. Turn off RDMA for PCI function 1
   ```
   niccli -i <index> nvm --setoption support_rdma --scope 1 --value 0
   ```
6. Use the following commands to determine if RDMA is enabled on the VFs:
   a. Query PCI function 0 to see if RDMA is enabled on child VFs
      ```
      niccli -i <index> nvm --getoption disable_rdma_srvio --scope 0
      ```
   b. Query PCI function 1 to see if RDMA is enabled on child VFs
      ```
      niccli -i <index> nvm --getoption disable_rmda_sriov --scope 1
      ```
7. Use the following commands to determine RDMA on VFs:
   a. Disable RDMA on PCI function 0 child VFs

```
niccli -i <index> nvm --setoption disable_rdma_sriov --scope 0 --value 1
```

b. Disable RDMA on PCI function 1 child VFs

```
niccli -i <index> nvm --setoption disable_rdma_sriov --scope 1 --value 1
```

c. Enable RDMA on PCI function 0 child VFs

```
niccli  -i <index> nvm --setoption disable_rdma_sriov --scope 0 --value 0
```

d. Enable RDMA on PCI function 1 child VFs

```
niccli -i <index> nvm --setoption disable_rdma_sriov --scope 1 --value 0
```

Enabling QoS for RDMA (Optional)

Enabling QoS features requires the Data Center Bridging (DCB) feature to be installed on the Windows Server operating system.

Installing DCB

Use the following PowerShell command to install DCB:

```
Install-WindowsFeature -Name Data-Center-Bridging -IncludeManagementTools
```

Use the following PowerShell commands to configure QoS. The command examples affect all network adapters. To target a specific NIC, append a `-Name` parameter to the PS command.

1. Use the following command to view the existing QoS configuration:
   ```
   Get-NetAdapterQos
   ```
2. Use the following command to enable QoS on all net adapters:
   ```
   Enable-NetAdapterQos -InterfaceAlias "Slot 5 Port 1"
   ```
3. Use the following command to disable the willing bit for Windows:
   ```
   Set-NetQosDcbxSetting -Willing $False
   ```
4. PFC is only supported on one Traffic Class (TC) at a time. Use the following command to disable other traffic classes:
   ```
   Disable-NetQosFlowControl -Priority 0,1,2,4,5,6,7
   ```
5. Use the following command to turn on PFC on priority 3:
   ```
   Enable-NetQosFlowControl -priority 3
   ```
6. Use the following command to view the updated priority configuration changes:
   ```
   Get-NetQosFlowControl
   ```
7. Use the following command to assign RDMA traffic to the TC named “MyTrafficClass.” SMBDirect uses port 445, however, the driver translates this classification to a RoCEv2 UDP port so this command will put all RDMA traffic on the assigned TC, not just SMBD traffic on port 445:
   ```
   New-NetQosPolicy "MyTrafficClass" –NetDirectPortMatchingCondition 445 –PriorityValue8021Action 3
   ```
8. Use the following command to create a new traffic class for RDMA named “MyTrafficClass”. Set the class to use priority 3 and give it 50% of the bandwidth. This command creates the TC, but does not assign any traffic.
   ```
   New-NetQosTrafficClass "MyTrafficClass" -Priority 3 -BandwidthPercentage 50 -Algorithm ETS
   ```
9. RDMA traffic is now configured on pri 3 with PFC enabled and with 50% ETS bandwidth reservation. The bandwidth reservation guarantees at least 50% of the available bandwidth to that traffic class (it can use more if available).
10. Use the following commands to remove the QoS configuration:
    ```
    Remove-NetQosPolicy -Confirm:$False
    Remove-NetQosTrafficClass
    Disable-NetQosFlowControl -Priority 0,1,2,3,4,5,6,7
    ```

Configuring Windows RoCE for Three Usage Types

There are three modes of RDMA support in Windows. Mode 1 is on the PF and does not involve a virtual switch. Mode 2 and Mode 3 require a vSwitch to be present on the system. The following sections provide configuration guidance for each RDMA mode that Windows Server supports.

Mode 1 RDMA

Mode 1 RDMA is one of three NIC operating modes. Mode 1 RDMA refers to using RDMA over the PF. This model is the most basic use model and is the result of performing the previous configuration steps. RDMA operates using the same interface as the Windows network stack. Mode 1 RDMA cannot be used together with mode 2 or mode 3. This is because mode 1 operates without a switch while mode 2 and mode 3 use a v-switch.

Mode 2 RDMA

Mode 2 RDMA refers to using RDMA over a virtual NIC (vNIC) on the host. Although it has the word virtual in the name, the vNIC is used by the host and does not use SR-IOV. A vNIC is an additional L2 interface with its own MAC and VLAN configuration that exists over the PF. This configuration requires the creation of a vSwitch. Each vNIC has its own instance of the Windows networking stack. Mode 2 vNICs can be used together on the same switch as Mode 3 vmNICs. Use the following steps to create vNIC interfaces:

1. Create a vSwitch and explicitly exclude the management OS at switch using the following command:
   ```
   New-VMSwitch -Name MySwitch -NetAdapterName “Slot 5 Port 1” -AllowManagementOS $False
   ```
2. Create three virtual NICs on top of this switch for the host OS using the following command:
   ```
   Add-VMNetworkAdapter -ManagementOS -Name vNIC1 -SwitchName MySwitch
   ```
3. Assign IP addresses to the vNIC by using the following command:
   ```
   New-NetIPAddress –InterfaceAlias "vEthernet (vNIC1)" –IPAddress “192.168.25.2” –PrefixLength 24
   ```
4. Enable RDMA on each vNIC by using the following commands:
   ```
   Enable-NetAdapterRdma -Name "vEthernet (vNIC1)"
   ```
5. If required, create each vNIC on its own VLAN by using the following command:
   ```
   Set-VMNetworkAdapterVlan -ManagementOS -Access -VlanId 12 -VMNetworkAdapterName vNIC1
   ```
6. Use the following command to remove VLAN tagging on a vNIC:
   ```
   Set-VMNetworkAdapterVlan -ManagementOS -Untagged -VMNetworkAdapterName vNIC1
   ```
7. Use the following command to enable Priority tagging on a vNIC:
   ```
   Set-VMNetworkAdapter -ManagementOS -Name vNIC1 -IeeePriorityTag on
   ```
8. Confirm the RDMA listeners are active using the following command:
   ```
   netstat -xan
   ```

Mode 3 RDMA

Mode 3 RDMA is the last RDMA mode. This mode refers to using RDMA in a VM using SR-IOV. This type of NIC is called a vmNIC as opposed to a vNIC for mode 2. The following steps are used to create a mode 3 NIC. Mode 2 and mode 3 NICs can be used together.

To create a VF with RDMA enabled, use the following PowerShell commands:

1. Create a switch on the PF adapter by using the following command:
   ```
   New-VMSwitch MySwitch -NetAdapterName "SLOT 5 Port 1"-EnableIov $true
   ```
2. Create a VF NIC for an existing VM named vm1 by using the following command:
   ```
   Add-VMNetworkAdapter -VMName vm1 -SwitchName MySwitch -Name nic1
   ```
3. Enable SR-IOV on the VF NIC by using the following command:
   ```
   Set-VMNetworkAdapter -VMName vm1 -VMNetworkAdapterName nic1 -IovWeight 100
   ```
4. Enable RDMA on the VF NIC by using the following command. This command gives the VF NIC RDMA capability, but RDMA must be enabled from within the VF.
   ```
   Set-VMNetworkAdapterRdma -VMName vm1 -VMNetworkAdapterName nic1 -RdmaWeight 100
   ```

5. Start the VM and install the Broadcom device driver.
6. Enable NetworkDirectFunctionality on the VF and NetworkDirect on the associated Microsoft Adapter.
7. Verify that SMBD listeners are present using the following command:

```
netstat -xan
```

User Mode RDMA

Broadcom NXE products support User Mode RDMA applications using the Microsoft NDSPI interface. Support for user mode applications is provided by Broadcom DLL `bxndspi.dll`.

Before running a user-mode application written to NDSPI, copy and install the `bxndspi.dll` user-mode driver.

1. Copy `bxndspi.dll` to `C:\Windows\System32`.
2. Install the driver by running the following command:

```
rundll32.exe .\bxndspi.dll,Config install|more
```

For questions regarding the MS NDSPI interface please refer to MS documentation found at: https://github.com/microsoft/NetworkDirect#networkdirect-spi

Verification

Use the Performance Monitor (Perfmon) or Power Shell to verify a RDMA connection.

- Perfmon
  Using Perfmon, the total number of active RDMA connections can be verified by using the RDMA Activity counter. Verification for which of the 16 channels are being exercised can be done by using the SMB direct connection counters.
  Note: Ensure that traffic is running to see the counters.

Figure 37: Perfmon for Verification (Part 1)

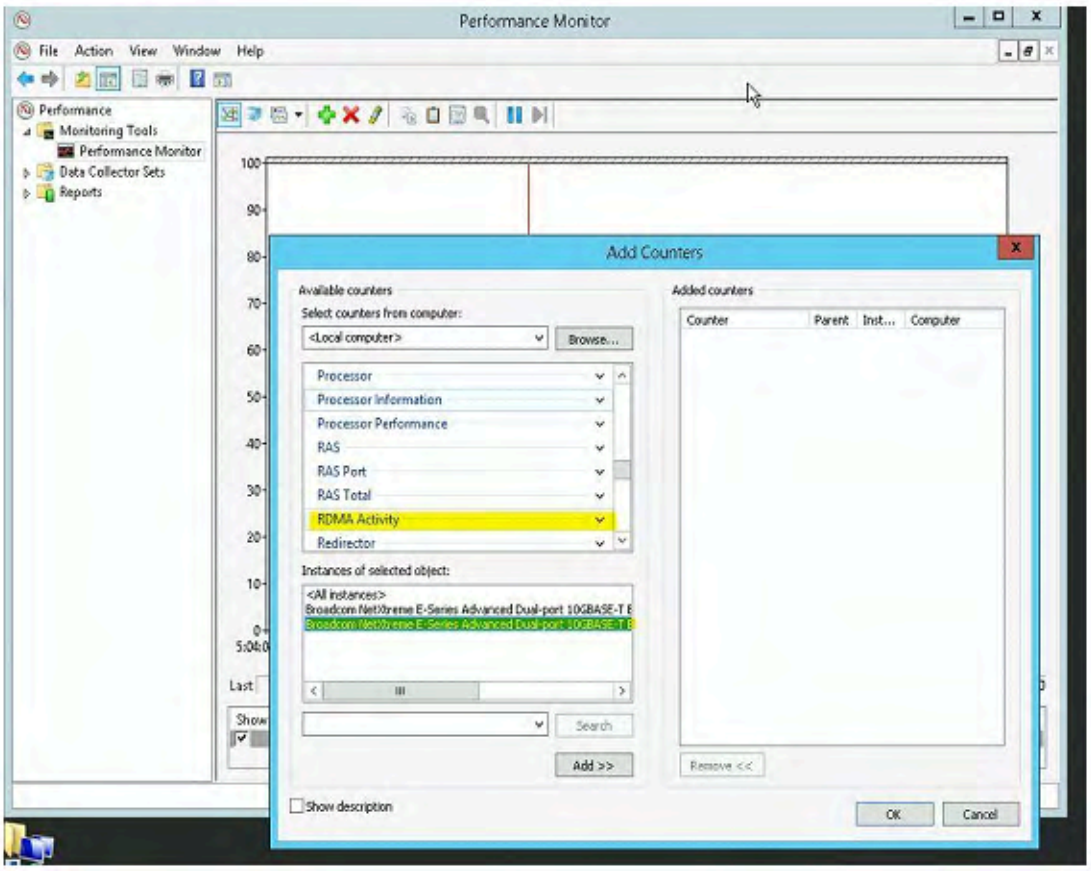


Figure 38: Perfmon for Verification (Part 2)

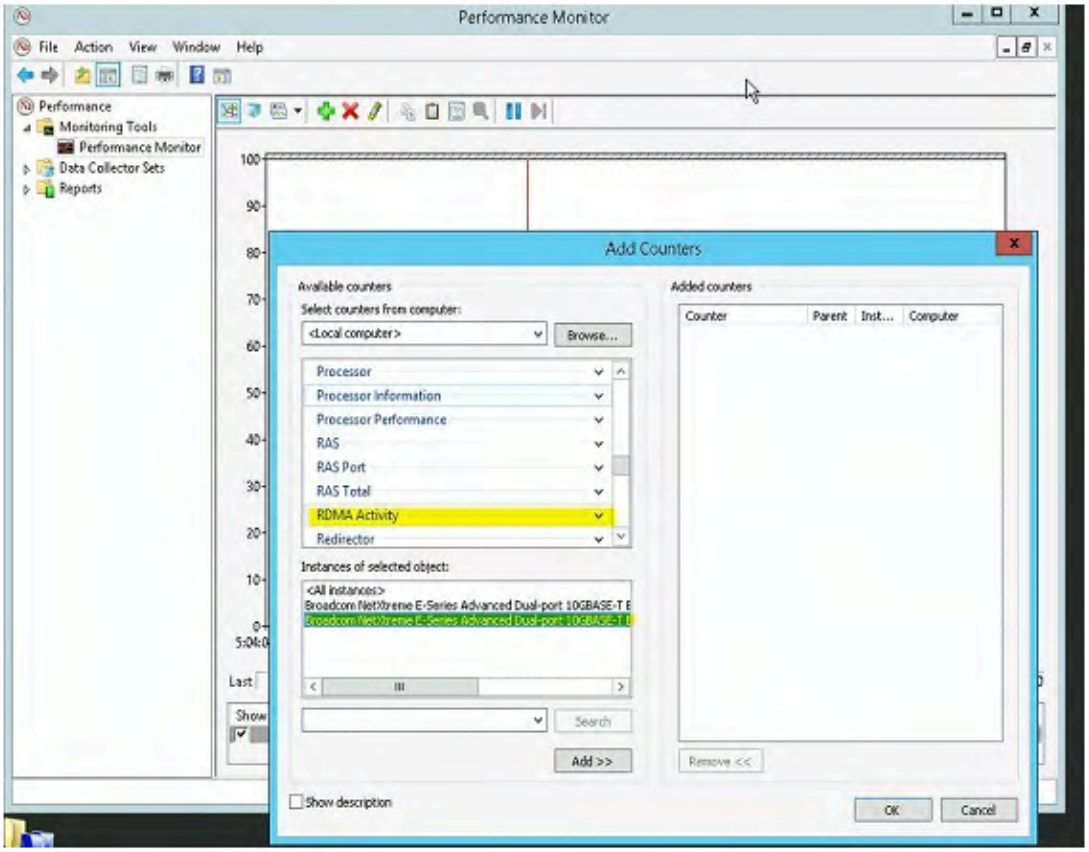
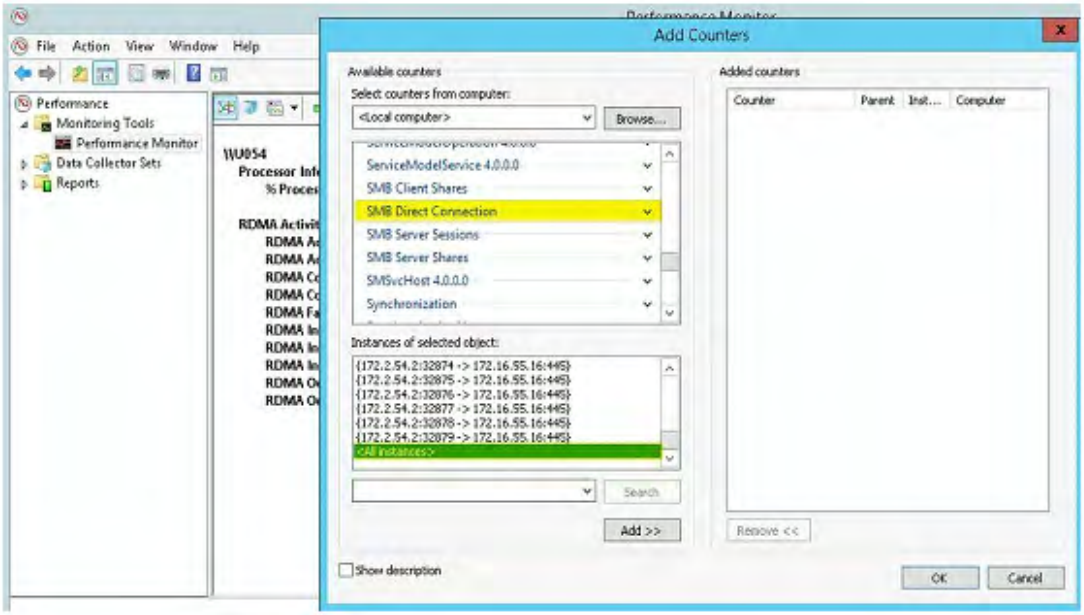
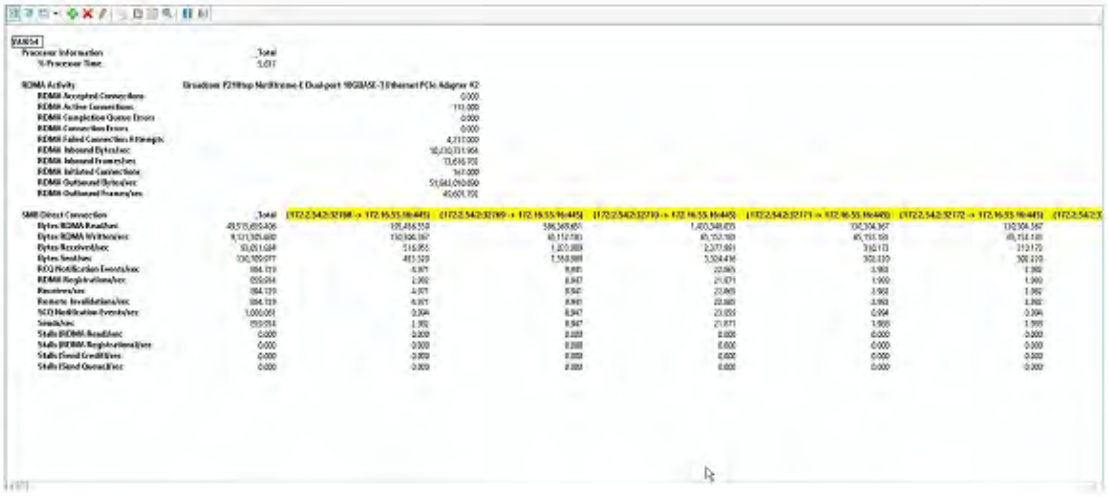


Figure 39: Perfmon for Verification (Part 3)



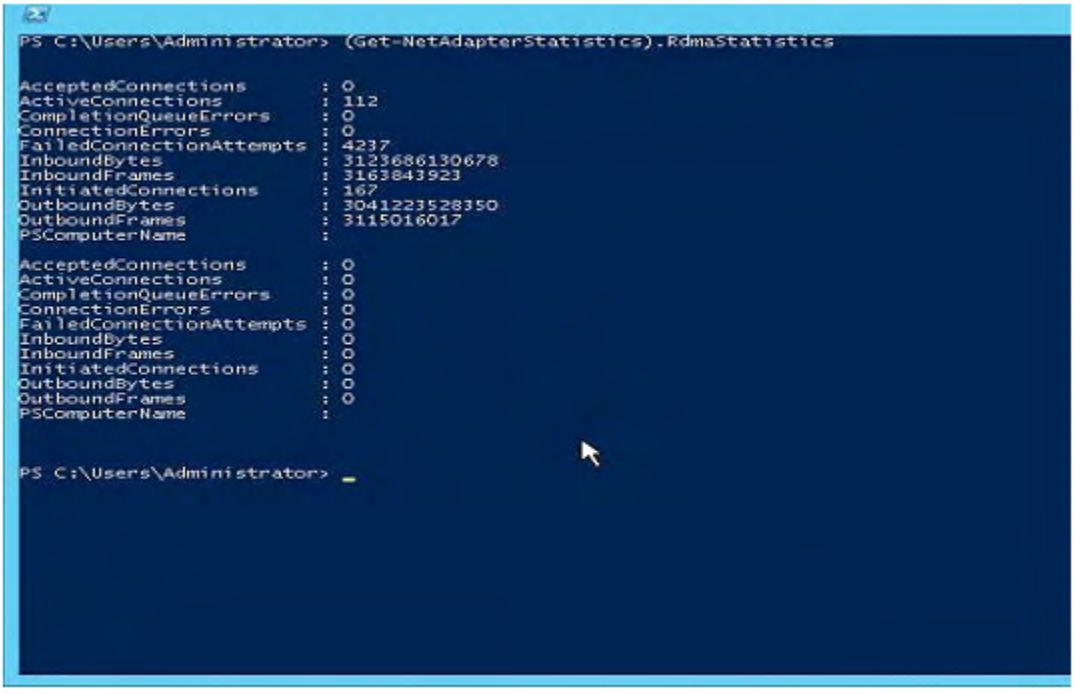


- Powershell

Use the following commands at the Powershell prompt. There are connections listed if using RDMA. Otherwise, it defaults to TCP.

- `Netstat -xan` – Shows RDMA connections
- `Netstat -an` – Shows TCP connections
- `(Get-NetAdapterStatistics). RDMAStatistics` – Lists RDMA Activity Connections

Figure 40: Powershell for Verification



Tuning RoCE for Ethernet Network Adapters

Provides tuning information to improve RoCE performance for Ethernet network adapters.

Congestion control is a comprehensive solution used for congestion management that can help reduce packet losses and congestion spreading as well as improve latency by keeping switch queue levels low.

Congestion control performance is measured by these network traffic performance metrics during periods of congestion:

- Fairness of bandwidth allocation between QPs
- Link utilization
- Latency

Under heavy congestion, congestion control can enforce fairness at a per-QP level, with low variation between streams. Streams not crossing the point of network contention are not affected.

Performance can vary depending on the topology, number of flows, traffic types, and the application used.
Use careful consideration and understanding when adjusting the different congestion control parameters. As a default, probablistic marking with the default CC algorithm in DCQCN-P is recommended and can be configured using the setup script provided in Congestion Control Tuning Parameters.

This section provides the following information on performance tuning:

- Disable CPU Power Saving
- Tuning Applications for High QP Count
- Improving RDMA Workload Performance

Disable CPU Power Saving

Provides information for disabling CPU power savings.

Intel CPU power-saving technology can cause RoCE packet drops with bursty traffic. It is highly recommended to set BIOS power saving mode to maximum performance and disable c-states and p-states with kernel options by editing the `/etc/ default/grub` file, and appending the following to `GRUB_CMDLINE_LINUX_DEFAULT` :

```
intel_pstate=disable processor.max_cstate=1 intel_idle.max_cstate=0
```

Rebuild the grub configuration as follows:

```
sudo /usr/sbin/grub-mkconfig* -o /boot/grub/grub.cfg
```

Tuning Applications for High QP Count

Provides information on setting the High QP count to minimize RNR-NAKs.

At larger scales, when hundreds of QPs are active and the Send/Recv protocol exchanges data, observing RNR-NAKs is possible. In accordance with the IB specification, RNR-NAKs are recoverable errors and the application can be tuned to minimize the occurrence of RNR-NAKs. Parameters that help minimize the RNR-NAKs are as follows:

- Bind the task to the CPU with a matching NUMA node to the network adapter.
- Increase the depth of RX queue using an application-specific parameter. For example, `ib_send_bw` has the -r option to increase receive queue depth.
- If the application allows, tune the threshold of completion suppression. This tuning is also known as CQ moderation. For example, `ib_send_bw` has the -Q option.

Improving RDMA Workload Performance

Provides information on improving RDMA workload performance.

Use the following recommendations for improving RDMA workload performance:

1. Refer to the recommendations for the best BIOS performance related to NUMA Per Socket (NPS), the preferred I/O device, and IOMMU.
2. For user-mode RDMA workloads, the recommended configuration for optimal performance is to ensure that the number of application processes/threads does not exceed the number of CPU cores. Typically, one application process is mapped to one CPU core in HPC and ML applications. Over-subscription of the application/core ratio may lead to decreased system efficiency due to higher cache misses and OS scheduling overheads.
3. The system configuration should also account for any co-existing non-RDMA or management application usage of the CPU cores.
4. Unless an application has been thoroughly tested in the hyper-threaded (HT) environment, it is recommended to disable hyper-threading.

Switch Configuration for RDMA over Converged Ethernet (RoCE)

Provides information on RoCE switch configuration.

Default NIC Parameters for Switch Configuration

The default NIC configuration enables the use of Explicit Congestion Notification (ECN), Priority Flow Control (PFC), and Enhanced Transmission Selection (ETS) on the network adapter. ECN and PFC are used because the RoCE protocol is network congestion sensitive. For congestion control purposes, the priority settings on the network must match the Congestion Control settings on the RNIC.

For a better understanding of ECN and PFC, see RoCE Congestion Control. The default RNIC settings enable RoCE packets to be marked with a DSCP value of 26 and the CNP packets to be marked with a DSCP value of 48. If VLAN Tagging is enabled, the RoCE packets are marked with a VLAN Priority Code Point (PCP) value of 3 and the CNP packets are marked with a VLAN PCP value of 7. PFC is enabled for Priority 3 traffic and ETS is configured between RoCE and non-ROCE traffic to be 50% each. The DSCP and VLAN priorities are also described in DSCP and VLAN.

See Viewing/Changing the RoCE DSCP/VLAN Settings for information on viewing and changing the RoCE DSCP/VLAN settings.

200G Link Speeds

See Configuring 200G Link Speeds on Ethernet Network Adapters.

ECN Thresholds

Congestion control uses Explicit Congestion Notification (ECN) (see RoCE Congestion Control) to react to marked packets within the network switch infrastructure during times of congestion. The correct ECN threshold value depends on the link speed of that port, and the congestion control protocol being used. See Deterministic Marking (DCQCN-D) and Probabilistic Marking (DCQCN-P) for a description of the available congestion control protocols. Use the protocol and link speed when selecting the correct threshold from the following tables for your switch configuration, then use the example switch configuration text below with updated ECN threshold values for your switch configuration.

Table 67: Mark Values (Deterministic DCQCN-D)

Switch Egress Port Link Speed (Gb/s)	Deterministic (DCQCN-D)		
	ECN Min Threshold (Kilobytes)	ECN Max Threshold (Kilobytes)	Probability
10	12	13	100%
25	16	17	100%
50	24	25	100%
100	64	65	100%
200	64	66	100%
400*	130	131	100%

* For interswitch (for example, spine-to-leaf).

Table 68: Mark Values (Probabilistic DCQCN-P)

Switch Egress Port Link Speed (Gb/s)	Probabilistic (DCQCN-P)		
	ECN Min Threshold (Kilobytes)	ECN Max Threshold (Kilobytes)	Probability
10	TBD	TBD	TBD
25	TBD	TBD	TBD
50	TBD	TBD	TBD
100	500	1500	20%
200	500	1500	20%
400*	1000	3000	20%

* For interswitch (for example, spine-to-leaf)

Each switch port participating in congestion control must be configured with the previous ECN threshold for the class of service associated with RoCEv2 traffic. Different vendors have different naming conventions that specify the minimum and maximum marking threshold and the marking percentage.

Switch Configuration Examples

The switch configuration elements required are as follows:

- Map DSCP traffic priorities for RoCE and CNP traffic to traffic classes
- Enable PFC
- Enable ECN
- Enable ETS

Example: Arista 7060CX (DCQCN-P at 100G)

The following example shows how to configure the required switch elements for the Arista switch.

```
qos map dscp 26 to traffic-class 3
qos map dscp 48 to traffic-class 7
!
```

```
interface Ethernet1/1
tx-queue 3
random-detect ecn minimum-threshold 500 kbytes maximum-threshold 1500 kbytes max-mark-probability 20
priority-flow-control mode on
priority-flow-control priority 3 no-drop
```

Example: Arista 7060CX (DCQCN-D at 100G)

The following example shows how to configure the required switch elements for the Arista switch.

```
qos map dscp 26 to traffic-class 3
qos map dscp 48 to traffic-class 7
!
interface Ethernet1/1
tx-queue 3
random-detect ecn minimum-threshold 64 kbytes maximum-threshold 65 kbytes max-mark-probability 100
priority-flow-control mode on
priority-flow-control priority 3 no-drop
```

Example: Dell Z9332F-ON (DCQCN-P at 100G)

The following example shows how to configure the required switch elements for the Dell Z9332F-ON switch.

```
wred ECN
random-detect color green minimum-threshold 500 maximum-threshold 1500 drop-probability 20
random-detect color yellow minimum-threshold 500 maximum-threshold 1500 drop-probability 20
random-detect color red minimum-threshold 500 maximum-threshold 1500 drop-probability 20
random-detect ecn
!
system qos
buffer-statistics-tracking
pfc-max-buffer-size 17408
pfc-shared-buffer-size 10240
!
wred mem
!
class-map type application class-iscsi
!
class-map type qos c3
!
class-map type qos c7
!
class-map type queuing Q0
match queue 0
!
class-map type queuing Q3
match queue 3
!
class-map type queuing Q7
match queue 7
!
class-map type network-qos cPFC3
match qos-group 3
!
```

```
trust dscp-map rDSCP
qos-group 7 dscp 48
qos-group 3 dscp 26
!
qos-map traffic-class tc3Q
queue 3 qos-group 3 type ucast
queue 7 qos-group 7 type ucast
queue 0 qos-group 0-2,4-6 type ucast
!
policy-map type application policy-iscsi
!
policy-map type network-qos PocPFC3
!
class cPFC3
pause
pfc-cos 3
!
policy-map type queuing po2Q
!
class Q0
!
class Q3
random-detect ECN
queue-limit thresh-mode dynamic 5
!
class Q7
!
port-group 1/1/1
profile unrestricted
port 1/1/1 mode Eth 100g-2x
port 1/1/2 mode Eth 100g-2x
!
port-group 1/1/2
profile unrestricted
port 1/1/3 mode Eth 100g-2x
port 1/1/4 mode Eth 100g-2x
!
port-group 1/1/3
profile unrestricted
port 1/1/5 mode Eth 100g-2x
port 1/1/6 mode Eth 100g-2x
!
port-group 1/1/4
profile unrestricted
port 1/1/7 mode Eth 100g-2x
port 1/1/8 mode Eth 100g-2x
!
port-group 1/1/5
profile unrestricted
port 1/1/9 mode Eth 100g-2x
port 1/1/10 mode Eth 100g-2x
!
port-group 1/1/6
```

```
profile unrestricted
port 1/1/11 mode Eth 100g-2x
port 1/1/12 mode Eth 100g-2x
!
port-group 1/1/7
profile unrestricted
port 1/1/13 mode Eth 100g-2x
port 1/1/14 mode Eth 100g-2x
!
port-group 1/1/8
profile unrestricted
port 1/1/15 mode Eth 100g-2x
port 1/1/16 mode Eth 100g-2x
!
port-group 1/1/9
profile unrestricted
port 1/1/17 mode Eth 100g-2x
port 1/1/18 mode Eth 100g-2x
!
port-group 1/1/10
profile unrestricted
port 1/1/19 mode Eth 100g-2x
port 1/1/20 mode Eth 100g-2x
!
port-group 1/1/11
profile unrestricted
port 1/1/21 mode Eth 100g-2x
port 1/1/22 mode Eth 100g-2x
!
port-group 1/1/12
profile unrestricted
port 1/1/23 mode Eth 100g-2x
port 1/1/24 mode Eth 100g-2x
!
port-group 1/1/13
profile unrestricted
port 1/1/25 mode Eth 100g-2x
port 1/1/26 mode Eth 100g-2x
!
port-group 1/1/14
profile unrestricted
port 1/1/27 mode Eth 100g-2x
port 1/1/28 mode Eth 100g-2x
!
port-group 1/1/15
profile unrestricted
port 1/1/29 mode Eth 100g-2x
port 1/1/30 mode Eth 100g-2x
!
port-group 1/1/16
profile unrestricted
port 1/1/31 mode Eth 100g-2x
port 1/1/32 mode Eth 100g-2x
```

```
!
interface vlan1
no shutdown
!
interface mgmt1/1/1
no shutdown
ip address dhcp
ipv6 address autoconfig
!
interface range ethernet1/1/1:1-1/1/32:5
no shutdown
switchport access vlan 1
mtu 9216
flowcontrol receive off
flowcontrol transmit off
priority-flow-control mode on
service-policy input type network-qos PocPFC3
service-policy output type queuing po2Q
qos-map traffic-class tc3Q
trust-map dscp rDSCP
```

Example: Dell Z9332F-ON (DCQCN-D at 100G)

The following example shows how to configure the required switch elements for the Dell Z9332F-ON switch.

```
wred ECN
random-detect color green minimum-threshold 64 maximum-threshold 65 drop-probability 100
random-detect color yellow minimum-threshold 64 maximum-threshold 65 drop-probability 100
random-detect color red minimum-threshold 64 maximum-threshold 65 drop-probability 100
random-detect ecn
!
system qos
buffer-statistics-tracking
pfc-max-buffer-size 17408
pfc-shared-buffer-size 10240
!
wred mem
!
class-map type application class-iscsi
!
class-map type qos c3
!
class-map type qos c7
!
class-map type queuing Q0
match queue 0
!
class-map type queuing Q3
match queue 3
!
class-map type queuing Q7
match queue 7
!
```

```
class-map type network-qos cPFC3
match qos-group 3
!
trust dscp-map rDSCP
qos-group 7 dscp 48
qos-group 3 dscp 26
!
qos-map traffic-class tc3Q
queue 3 qos-group 3 type ucast
queue 7 qos-group 7 type ucast
queue 0 qos-group 0-2,4-6 type ucast
!
policy-map type application policy-iscsi
!
policy-map type network-qos PocPFC3
!
class cPFC3
pause
pfc-cos 3
!
policy-map type queuing po2Q
!
class Q0
!
class Q3
random-detect ECN
queue-limit thresh-mode dynamic 5
!
class Q7
!
port-group 1/1/1
profile unrestricted
port 1/1/1 mode Eth 100g-2x
port 1/1/2 mode Eth 100g-2x
!
port-group 1/1/2
profile unrestricted
port 1/1/3 mode Eth 100g-2x
port 1/1/4 mode Eth 100g-2x
!
port-group 1/1/3
profile unrestricted
port 1/1/5 mode Eth 100g-2x
port 1/1/6 mode Eth 100g-2x
!
port-group 1/1/4
profile unrestricted
port 1/1/7 mode Eth 100g-2x
port 1/1/8 mode Eth 100g-2x
!
port-group 1/1/5
profile unrestricted
port 1/1/9 mode Eth 100g-2x
```

```
port 1/1/10 mode Eth 100g-2x
!
port-group 1/1/6
profile unrestricted
port 1/1/11 mode Eth 100g-2x
port 1/1/12 mode Eth 100g-2x
!
port-group 1/1/7
profile unrestricted
port 1/1/13 mode Eth 100g-2x
port 1/1/14 mode Eth 100g-2x
!
port-group 1/1/8
profile unrestricted
port 1/1/15 mode Eth 100g-2x
port 1/1/16 mode Eth 100g-2x
!
port-group 1/1/9
profile unrestricted
port 1/1/17 mode Eth 100g-2x
port 1/1/18 mode Eth 100g-2x
!
port-group 1/1/10
profile unrestricted
port 1/1/19 mode Eth 100g-2x
port 1/1/20 mode Eth 100g-2x
!
port-group 1/1/11
profile unrestricted
port 1/1/21 mode Eth 100g-2x
port 1/1/22 mode Eth 100g-2x
!
port-group 1/1/12
profile unrestricted
port 1/1/23 mode Eth 100g-2x
port 1/1/24 mode Eth 100g-2x
!
port-group 1/1/13
profile unrestricted
port 1/1/25 mode Eth 100g-2x
port 1/1/26 mode Eth 100g-2x
!
port-group 1/1/14
profile unrestricted
port 1/1/27 mode Eth 100g-2x
port 1/1/28 mode Eth 100g-2x
!
port-group 1/1/15
profile unrestricted
port 1/1/29 mode Eth 100g-2x
port 1/1/30 mode Eth 100g-2x
!
port-group 1/1/16
```

```
profile unrestricted
port 1/1/31 mode Eth 100g-2x
port 1/1/32 mode Eth 100g-2x
!
interface vlan1
no shutdown
!
interface mgmt1/1/1
no shutdown
ip address dhcp
ipv6 address autoconfig
!
interface range ethernet1/1/1:1-1/1/32:5
no shutdown
switchport access vlan 1
mtu 9216
flowcontrol receive off
flowcontrol transmit off
priority-flow-control mode on
service-policy input type network-qos PocPFC3
service-policy output type queuing po2Q
qos-map traffic-class tc3Q
trust-map dscp rDSCP
```

If the default RNIC settings are not appropriate for a particular network, the Broadcom tools: `niccli` and `bnxtsetupcc.sh` can be used to modify the default settings. The RNIC settings should, however, match the network switch settings. See RDMA SR-IOV for Ethernet Network Adaptersfor more information on how to modify the RNIC settings.

Supporting Multiple Lossless Queues and Traffic Classes

Provides information on supporting, configuring, and utilizing multiple lossless traffic classes.

Support for multiple lossless traffic classes allows the user to run multiple independent RoCE applications, each with its distinct QoS characteristics.

Configuring Multiple Lossless Traffic Classes

By default, only a single traffic class is configured as lossless - traffic class 1. The user can utilize the NICCLI tools to change which traffic classes are configured as lossless. This can be done both at runtime and persistently, or across reboots. Only traffic classes 3 through 7 can be changed from lossless to lossy and vice versa. Traffic classes 0 and 2 must remain lossy, while traffic class 1 must be configured as lossless.

NICCLI command to configure traffic classes 1, 5, 6, and 7 as lossless at run-time:

```
niccli -i 1 qos --ingress --cosq --set --mode 226
```

NICCLI command to configure traffic classes 1, 5, 6, and 7 as lossless persistently:

```
niccli -i 1 nvm --setoption cosq_mode_valid --value 1
niccli -i 1 nvm --setoption cosq_mode --value 226
```

A reboot is required for the configuration to remain persistent.

NICCLI command to retrieve which traffic classes are configured as lossless:

```
niccli -i 1 qos ingress --cosq --show
```

Utilizing Multiple Lossless Traffic Classes

Multiple lossless traffic classes can be exposed to the user through either perftest-based tools or through the uverbs interface.

Selecting One of Multiple Lossless Traffic Classes Using Perftest-Based Tools

Selecting which lossless traffic class should be used by a perftest based application can be done by specifying either the "--tclass" or "--sl" option, as specified by the client.

Perftest command to select the lossless traffic class based on the "--tclass" (DSCP) value of 38

Server:

```
ib_write_bw --report_gbits -F -d bnxt_re0  -x 3 -u 18 -q 4096 --run_infinitely -p 3003
```

Client:

```
ib_write_bw --report_gbits -F -d bnxt_re0  -x 3 -u 18 -q 4096 --run_infinitely -p 3003 40.1.1.2 --tclass=152
```

"--tclass" is DSCP (38) left shifted by 2. DSCP value has to map to a priority, that maps to a lossless queue.

Perftest command to select the lossless traffic class based on "--sl" (vlan pri or "service level") value of 6

Server:

```
b_write_bw --report_gbits -F -d bnxt_re0  -x 3 -u 18 -q 4096 --run_infinitely -p 3003
```

Client:

```
ib_write_bw --report_gbits -F -d bnxt_re0  -x 3 -u 18 -q 4096 --run_infinitely -p 3003 40.1.1.2 --sl=6
```

"Service level" value has to map to a priority, that maps to a lossless queue. Both "--tclass" and "--sl" options can be used in the same command. The constraint here is that both the dscp value and the service level have to map to the same lossless queue. If a user incorrectly chooses to send RoCE traffic on a traffic class configured as lossy, the RoCE driver overrides the user and sends it with the DSCP and service level of the default RoCE traffic class (traffic class 1). If there's no DSCP mapped to a priority / traffic class that's being used, no DSCP is specified in the packet. DSCP-to-priority and priority-to-cos information is used when RoCE traffic is launched. This information is used for the lifetime of the QP. Even if one of those two mappings were to change after traffic is launched, updating this configuration would not change the mapping for the existing traffic and does not effect already running traffic.

Selecting One of Multiple Lossless Traffic Classes Using the uverbs Interface

Selection of lossless traffic class can be done by using the ibv_modify_qp() uverb API by passing the "traffic_class" and "sl" values. When the QP moves from IBV_QPS_INIT to IBV_QPS_RTR state, these parameters should be passed to ibv_modify_qp() through the attributes parameter.

Example:

Select the lossless queue based on the dscp value 38.

```
attr->ah_attr.grh.traffic_class = 152;  /* traffic class is dscp left shifted by 2 */
```

Select the lossless queue based on the "service level" value of 6.

Backward Compatibility

Multiple lossless traffic classes are only supported when both the RoCE driver and firmware support this feature. Otherwise, both the firmware and driver default to using the only lossless traffic class that was supported previously, which is traffic class 1.

Retrieving RoCE Statistics

Provides information on using the sysfs files to retrieve RoCE statistics.

ethtool Statistics for all Traffic Types

The following example shows an example of ethtool statistics using enp8s0f0np0 as the interface name.

```
# ethtool -S enp8s0f0np0  | grep -i cos
    rx_bytes_cos0: 377976325178
    rx_packets_cos0: 349384543
    rx_bytes_cos1: 0
    rx_packets_cos1: 0
    rx_bytes_cos2: 0
    rx_packets_cos2: 0
    rx_bytes_cos3: 0
    rx_packets_cos3: 0
    rx_bytes_cos4: 82704
    rx_packets_cos4: 756
    rx_bytes_cos5: 1512
    rx_packets_cos5: 8
    rx_bytes_cos6: 0
    rx_packets_cos6: 0
    rx_bytes_cos7: 0
    rx_packets_cos7: 0
    rx_discard_bytes_cos0: 0
    rx_discard_packets_cos0: 0
    rx_discard_bytes_cos1: 0
    rx_discard_packets_cos1: 0
    rx_discard_bytes_cos2: 0
    rx_discard_packets_cos2: 0
    rx_discard_bytes_cos3: 0
    rx_discard_packets_cos3: 0
    rx_discard_bytes_cos4: 0
    rx_discard_packets_cos4: 0
    rx_discard_bytes_cos5: 0
    rx_discard_packets_cos5: 0
    rx_discard_bytes_cos6: 0
    rx_discard_packets_cos6: 0
    rx_discard_bytes_cos7: 0
    rx_discard_packets_cos7: 0
    tx_bytes_cos0: 411882014984
    tx_packets_cos0: 382108196
    tx_bytes_cos1: 0
    tx_packets_cos1: 0
    tx_bytes_cos2: 0
    tx_packets_cos2: 0
    tx_bytes_cos3: 0
    tx_packets_cos3: 0
    tx_bytes_cos4: 86526
    tx_packets_cos4: 767
    tx_bytes_cos5: 0
    tx_packets_cos5: 0
```

```
     tx_bytes_cos6: 0
     tx_packets_cos6: 0
     tx_bytes_cos7: 0
     tx_packets_cos7: 0
```

RoCE Statistics using Linux debugfs

Note: The following example was captured from a BCM5750X device.

RoCE-specific statistics, including congestion control statistics, can be viewed using the Linux sysfs interface of the RoCE interface.

The following example shows an example of RoCE statistics from the sysfs interface using bnxt_re0 as the RoCE interface name.

```
# cat /sys/kernel/debug/bnxt_re/bnxt_re0/info

[root@stx1 ~]# cat /sys/kernel/debug/bnxt_re/bnxt_re0/info
bnxt_re debug info:
=====[ IBDEV bnxt_re0 ]============================
       link state: UP
       Max QP:         65537
       Max SRQ:        4096
       Max CQ:         65536
       Max MR:         262144
       Max MW:         262144
       Max AH:         65536
       Max PD:         65536
       Active QP:      2
       Active RC QP:   0
       Active UD QP:   0
       Active DMABUF MR: 0
       Active HW AH:   0
       DMABUF MR Watermark: 0
       AH HW Watermark: 0
       Active SRQ:     0
       Active CQ:      2
       Active MR:      6
       Active MW:      2
       Active AH:      0
       Active PD:      4
       QP Watermark:   2
       SRQ Watermark:  0
       CQ Watermark:   2
       MR Watermark:   6
       MW Watermark:   2
       AH Watermark:   1
       PD Watermark:   4
       Resize CQ count: 0
       Recoverable Errors: 0
       Rx Pkts: 324283
       Rx Bytes: 26169638
       Tx Pkts: 324286
       Tx Bytes: 23576588
```

```
CNP Tx Pkts: 0
CNP Rx Pkts: 0
RoCE Only Rx Pkts: 324283
RoCE Only Rx Bytes: 26169638
RoCE Only Tx Pkts: 324286
RoCE Only Tx Bytes: 23576588
rx_roce_error_pkts: 0
rx_roce_discard_pkts: 0
tx_roce_error_pkts: 0
tx_roce_discards_pkts: 0
res_oob_drop_count: 0
tx_atomic_req: 0
rx_atomic_req: 0
tx_read_req: 0
tx_read_resp: 54040
rx_read_req: 54040
rx_read_resp: 0
tx_write_req: 0
rx_write_req: 54039
tx_send_req: 108080
rx_send_req: 108079
rx_good_pkts: 324283
rx_good_bytes: 26169638
rx_ecn_marked_pkts: 0
to_retransmits: 0
seq_err_naks_rcvd: 0
max_retry_exceeded: 0
rnr_naks_rcvd: 0
missing_resp: 0
unrecoverable_err: 0
bad_resp_err: 0
local_qp_op_err: 0
local_protection_err: 0
mem_mgmt_op_err: 0
remote_invalid_req_err: 0
remote_access_err: 0
remote_op_err: 0
dup_req: 0
res_exceed_max: 0
res_length_mismatch: 0
res_exceeds_wqe: 0
res_opcode_err: 0
res_rx_invalid_rkey: 0
res_rx_domain_err: 0
res_rx_no_perm: 0
res_rx_range_err: 0
res_tx_invalid_rkey: 0
res_tx_domain_err: 0
res_tx_no_perm: 0
res_tx_range_err: 0
res_irrq_oflow: 0
res_unsup_opcode: 0
res_unaligned_atomic: 0
```

```
res_rem_inv_err: 0
res_mem_error64: 0
res_srq_err: 0
res_cmp_err: 0
res_invalid_dup_rkey: 0
res_wqe_format_err: 0
res_cq_load_err: 0
res_srq_load_err: 0
res_tx_pci_err: 0
res_rx_pci_err: 0
res_oos_drop_count: 0
num_irq_started : 2
num_irq_stopped : 1
poll_in_intr_en : 0
poll_in_intr_dis : 0
cmdq_full_dbg_cnt : 0
fw_service_prof_type_sup : 1
dbq_int_recv: 0
dbq_int_en: 2
dbq_pacing_resched: 0
dbq_pacing_complete: 1
dbq_pacing_alerts: 0
dbq_dbr_fifo_reg: 0x56000000
dbr_drop_recov_epoch: 0
dbr_drop_recov_events: 0
dbr_drop_recov_timeouts: 0
dbr_drop_recov_timeout_users: 0
dbr_drop_recov_event_skips: 0
latency_slab [0 - 1] sec = 165
```

Note: Incrementing congestion notification packet (CNP) counts indicates congestion is detected within the network switch fabric.

RoCE Per QP Statistics

The driver can provide basic information about the active QPs. To gather the information, use the following command:

```
cat /sys/kernel/debug/bnxt_re/<pci bus:device:fn>/qp_info/<qp_id>
```

Example:

```
cat /sys/kernel/debug/bnxt_re/0000:af:00.0/qp_info/0x682
type = IB_QPT_RC(2)
state = IB_QPS_RTS(3)
source qpn = 1666
dest qpn = 1665
source port = 15750
dest port = 4791
port = 0
source_ipaddr = 11.11.11.3
destination_ipaddr = 11.11.11.1
timeout = 0
shaper allocated = 0
rate limit = 0 kbps
```

RoCE Counter Definitions

Provides definitions for various statistics for Ethernet network adapters.

The following table provides information on some of the RoCE-related statistics.

Table 69: Device Resource Limits

Counters	Description
`Max QP`	Maximum QP limit.
`Max SRQ`	Maximum SRQ limit.
`Max CQ`	Maximum CQs limit.
`Max MR`	Maximum memory region limit.
`Max MW`	Maximum memory window limit.
`Max AH`	Maximum Address handle limit.
`Max PD`	Maximum Protection domain limit.

Table 70: Active Resource Limits

Counters	Description
`Active QP`	Number of active QPs.
`Active SRQ`	Number of active SRQs.
`Active CQ`	Number of active CQs.
`Active MR`	Number of active memory regions.
`Active MW`	Number of active memory windows.
`Active AH`	Number of active address handles.
`Active PD`	Number of active Protection domains.
`Active RC QP`	Number of active RC queue pairs.
`Active UD QP`	Number of active UD queue pairs.
`Active DMABUF MR`	Number of active DMABUF Memory Regions.
`Active HW AH`	Number of active HW address handles.
`DMABUF MR Watermark`	Max DMABUF MRs active after driver load
`AH HW Watermark`	Max HW active after driver load.

Table 71: Resource Watermarks

Counters	Description
QP Watermark	Max QPs active after driver load.
SRQ Watermark	Max SRQs active after driver load.
CQ Watermark	Max CQs active after driver load.
MR Watermark	Max MRs active after driver load.
MW Watermark	Max MWs active after driver load.
AH Watermark	Max AHs active after driver load.

Counters	Description
PD Watermark	Max PDs active after driver load.

Table 72: Congestion Notification Counters

Counters	Description
CNP Tx Pkts	Number of RoCE CNP packets transmitted
CNP Rx Pkts	Number of RoCE CNP packets received
rx_ecn_marked_pkts	Number of ECN-marked RoCE packets received

Table 73: Transmit Counters

Counters	Description

`Tx_good_bytes`
Or
`RoCE Only Tx Bytes`
	Number of good RoCE bytes transmitted by hardware. This value includes UD and RC traffic for all protocols.

`Tx_good_pkts`
Or
`RoCE Only Tx Pkts`
	Number of good RoCE packets transmitted by hardware. This value includes UD and RC traffic for all protocols.

`tx_atomic_req`	Number of transmitted Atomic operation requests.
`tx_read_req`	Number of transmitted read request.
`tx_read_resp`	Number of transmitted read responses.
`tx_write_req`	Number of RDMA-write requests transmitted.
`tx_send_req`	Number of send requests transmitted.

Table 74: Receive Counters

Counters	Description

`Rx_good_bytes`
Or
`RoCE Only Rx Bytes`
Or
`Rx Bytes`
	Number of good RoCE bytes received by hardware, excluding errors and drops. This value includes UD and RC traffic including Acks and Naks for all protocols.

`Rx_good_pkts`
Or
`RoCE Only Rx Pkts`
Or
`Rx Pkts`
	Number of good RoCE packets received by hardware, excluding errors and drops. This value includes UD and RC traffic including Acks and Naks for all protocols.

`rx_atomic_req`	Number of atomic-operation-request received.
`rx_read_req`	Number of read requests received.
`rx_read_resp`	Number of read responses received.
`rx_write_req`	Number of RDMA write requests received.
`rx_send_req`	Number of incoming sends.

Table 75: Recoverable Errors

Counters	Description
Recoverable Errors	Number of recoverable errors detected. Recoverable errors are detected by the hardware. Hardware instructs FW to initiate the recovery process. The RC connection does not tear-down as a result of these errors.
to_retransmits	Number of retransmission requests because of ack timeouts.
rnr_naks_rcvd	Number of receiver-not-ready (RNR) NAKs received.
dup_req	Number of duplicated requests detected.
missing_resp	Number of responses missing.
rx_roce_error_pkts	Number of incoming packets dropped by hardware.
rx_roce_discard_pkts	Number of incoming packets discarded by hardware due to malformed packets.
res_oob_drop_count	Number of times the receiver engine depleted of message buffers (MBUFs).
res_oos_drop_count	Number of out-of-sequence RoCE packets received.

Table 76: Fatal Errors (Requester)

Counters	Description
seq_err_naks_rcvd	Number of PSN sequencing error NAKs received.
max_retry_exceeded	Number of retransmission requests exceeded the maximum.
unrecoverable_err	Number of unrecoverable errors detected.
bad_resp_err	Number of bad response errors detected.
local_qp_op_err	Number of QP local operation errors detected.
local_protection_err	Number of local protection errors detected.
mem_mgmt_op_err	Number of times H/W detected an error because of illegal bind/fast register/invalidate attempts
remote_invalid_req_err	Number of invalid requests received from the remote RDMA initiator.
remote_access_err	Number of times H/W received a REMOTE ACCESS ERROR NAK from the peer.
remote_op_err	Number of times H/W received a REMOTE OPERATIONAL ERROR NAK from the peer.

Table 77: Fatal Errors (Responder)

Counters	Description
res_exceed_max	Number of times H/W detected incoming send, RDMA write, or RDMA read messages which exceed the maximum transfer length.
res_length_mismatch	Number of times H/W detected incoming RDMA write message payload size does not match write length in the RETH.
res_exceeds_wqe	Number of times H/W detected send payload exceeds RQ/SRQ RQE buffer capacity.
res_opcode_err	Number of times H/W detected first, only, middle, last packets for incoming requests are improperly ordered with respect to the previous packet.
res_rx_invalid_rkey	Number of times H/W detected an incoming request with an R_KEY that did not reference a valid memory read/memory write command.

Counters	Description
res_rx_domain_err	Number of times H/W detected an incoming request with an R_KEY that referenced a MR/MW that was not in the same PD as the QP on which the request arrived.
res_rx_no_perm	Number of times H/W detected an incoming RDMA write request with an R_KEY that referenced a MR/MW which did not have the access permission needed for the operation.
res_rx_range_err	Number of times H/W detected an incoming RDMA write request that had a combination of R_KEY, VA, and length that was out of bounds of the associated MR/MW.
res_tx_invalid_rkey	Number of times H/W detected a R_KEY that did not reference a valid MR/MW while processing incoming read request.
res_tx_domain_err	Number of times H/W detected an incoming request with an R_KEY that referenced a MR/MW that was not in the same PD as the QP on which the RDMA read request is received.
res_tx_no_perm	Number of times H/W detected an incoming RDMA read request with an R_KEY that referenced a MR/ MW which did not have the access permission needed for the operation.
res_tx_range_err	Number of times H/W detected an incoming RDMA read request that had a combination of R_KEY, VA, and length that was out of bounds of the associated MR/MW.
res_irrq_oflow	Number of times H/W detected that peer sent more RDMA read or atomic requests than the negotiated maximum.
res_unsup_opcode	Number of times H/W detected that peer sent a request with an opcode for a request type that is not supported on this QP.
res_unaligned_atomic	Number of times H/W detected that VA of an atomic request is on a memory boundary that prevents atomic execution.
res_rem_inv_err	Number of times H/W detected an incoming send with invalidate request in which the R_KEY to invalidate did not MR/MW which could be invalidated.
res_mem_error64	Number of times H/W detected a RQ/SRQ SGE which points to an inaccessible memory.
res_srq_err	Number of times H/W detected a QP moving to error state because the associated SRQ is in error.
res_cmp_err	Number of times H/W detected that no CQE space is available on CQ or CQ is not in valid state.
res_invalid_dup_rkey	Number of times H/W detected invalid R_KEY while resending responses to duplicate read requests.
res_wqe_format_err	Number of times H/W detected error in the format of the WQE in the RQ/SRQ.
res_cq_load_err	Number of times H/W detected error while attempting to load the CQ context.
res_srq_load_err	Number of times H/W detected error while attempting to load the SRQ context.

Table 78: PCIe Errors

Counters	Description
res_tx_pci_err	Number of PCI errors detected during transmission by responder.
res_rx_pci_err	Number of PCI errors detected during reception by responder.

Table 79: Device Control Plane Counters

Counters	Description
num_irq_started	Number of times the control plane interrupt service routine had enabled.
num_irq_stopped	Number of times the control plane interrupt service routine had disabled.
poll_in_intr_en	Number of times a control path command had timed out in interrupt mode and the driver had to fall on to polling mode.
poll_in_intr_dis	Number of times control path command has been completed in pure polling mode because the interrupt mode was disabled.

Table 80: Driver Debug Counters

Counters	Description
Resize CQ count	Debug counter for CQ resize ops after driver load.
num_irq_started	Debug counter for IRQs started after device creation.
num_irq_stopped	Debug counter for IRQs stopped after device creation.
poll_in_intr_en	Debug counter for indicating control path polling when interrupt enabled.
poll_in_intr_dis	Debug counter for indicating control path polling when interrupts are disabled.
cmdq_full_dbg_cnt	Debug counter to indicate control path CMDQ full.
fw_service_prof_type_sup	Debug info to indicate the current service profile config.
dbq_int_recv	Debug counter to indicate the DBQ interrupt received.
dbq_int_en	Debug counter to indicate the number of iterations dbq interrupt is enabled.
dbq_pacing_resched	Debug counter to indicate the number of times pacing thread rescheduled.
dbq_pacing_complete	Debug counter to indicate the count where the pacing thread completed.
dbq_pacing_alerts	Debug counter to indicate how many times the library indicated the driver to start DB pacing.
dbq_dbr_fifo_reg	Debug counter to monitor the HW FIFO reg.
dbr_drop_recov_epoch	Debug counter to indicate epoch of latest DBR drop event.
dbr_drop_recov_events	Debug counter to indicate the number of DBR drop events.
dbr_drop_recov_timeouts	Debug counter to indicate the DBR drop events scheduled to the user space and failed to complete within the timeout.
dbr_drop_recov_timeout_users	Debug counter to indicate the number of user instances that experienced timeout when driver finishes the recovery thread.
dbr_drop_recov_event_skips	Debug counter to indicate the number of DBR drop events ignored (skipped) by the driver because of one or more outstanding event.
latency_slab	Each slab is of 1 second granularity. The Counters of each slab represent the total number of control path commands completed in that range. Up to 128 seconds latency is tracked.

For additional information, see RoCE Statistics.

Example RoCE + TCP Configuration on Ethernet Network Adapters

Provides an example of RoCE and TCP network configuration on Ethernet network adapters.

This section presents an example construction of a set of servers on a 100G network with RoCE enabled and configured for coexistence with IP traffic, using CentOS 8.3, and an Arista switch. This example is a typical pattern for a high-performance computing or machine-learning cluster. We will then test the cluster with the OSU Benchmark suite.

Downloading Software

Using a web browser, start with the Broadcom Ethernet Network Adapters page, select your product, go to the Downloads tab, Drivers folder, and download the Linux Driver Installer package.

Copy this package to each of the servers.

Installing RHEL

Install RHEL 9 from ISO or provisioning system using Minimal profile. Set the root password consistently.

Configuring the SSH Keys

On one server, which will be the control node, run the following commands:

```
# ssh-keygen
# eval `ssh-agent`
# ssh-copy-id root@<each other host>
```

Updating the Servers

On each server, run the following commands as root:

```
# yum update -y
# yum install gcc gcc-c++ make
# reboot
```

Installing the Software

On each server, run the following commands as root:

```
# tar xf <Linux Driver Installer Package File>
# cd <DIR>/Linux/Linux_Installer
# lspci | grep Broadcom
# bash install.sh -v -i <device PCIe address, eg: 41:00.0>
```

Note: Add `-a <IP> -n <NETMASK>` if the NIC does not already have a configured IP address.

Configuring the Switch

Configure the switch for PFC and ECN. In the following example, all ports are configured on a 32-port switch.

```
$ ssh admin@<Managemen IP>
$ enable
# configure terminal
# qos map dscp 26 to traffic-class 3
# qos map dscp 48 to traffic-class 7
# interface Ethernet 1/1-32/1
# speed 100g
# tx-queue 3
```

```
# random-detect ecn minimum-threshold 500 kbytes maximum-threshold 1500 kbytes max-mark-probability 20

# priority-flow-control mode on
```

Installing and Configuring OpenMPI

```
# sed -i 's/0x16f0,0x16f1/0x16f0,0x16f1,0x1750/' <ompi-install-path>/share/openmpi/mca-btl-openib-de-
vice-params.ini
```

Installing and Configuring UCX

On each server, run the following commands as root:

```
# wget https://github.com/openucx/ucx/releases/download/v1.12.1/ucx-1.12.1.tar.gz
# tar xzf ucx-1.15.x.tar.gz
# cd ucx-1.15.x
# ./contrib/configure-release --prefix=$PWD/install
# make -j8
# make install
```

OpenMPI versions less than 3.0 do not support UCX. It is recommended to install the latest version of OpenMPI.

```
# wget https://download.open-mpi.org/release/open-mpi/v4.1/openmpi-4.1.3.tar.gz
# tar zxf openmpi-4.1.3.tar.gz
# cd openmpi-4.1.1/
# ./autogen.pl
# mkdir build-ucx
# cd build-ucx
# ../configure --prefix=<ompi-install-path> --with-ucx=<ucx-install-path> --enable-mca-no-build=btl-uct
# sed -i 's/0x16f0,0x16f1/0x16f0,0x16f1,0x1750/' <ompi-install-path>/share/openmpi/mca-btl-openib-de-
vice-params.ini
```

By default, UCX tries to use all available transports and selects the best ones according to their performance capabilities and scale. It is possible to restrict the transports in use by setting `UCX_TLS=<t11>,<t12>` , and so forth, in the `mpirun` test command. Restricting the transport to `rc_verbs` as the scale of nodes and processors per node increases can result in test failures related to the creation of RC QPs when the QP limits supported in a given release are exceeded. It is recommended to avoid explicitly setting this variable in the test command and allowing UCX to select the transport automatically based on scaling requirements.

Compiling OSU Benchmarks

On each server, run the following commands as root:

```
On each server run the following commands as root:

# wget http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.7.tar.gz
# tar xf osu-micro-benchmarks-5.7.tar.gz
# cd osu-micro-benchmarks-5.7
# ./configure CC=/usr/bin/mpicc CXX=/usr/bin/mpicxx
# make
```

Running OSU Benchmarks

On the control node, server run the following command:

```
# /usr/bin/mpirun --allow-run-as-root -np 2 -host smc1,smc2 /root/osu-micro-benchmarks-5.7/mpi/pt2pt/
```

Windows Configuration Information

Provides information on configuring various Ethernet Adapter features in the Windows operating system.

The following list provides common Windows configuration options:

- Supported Operating Systems for Ethernet Network Adapters
- Installing the Windows Driver on Ethernet Network Adapters
- Updating the Firmware on Windows
- Installing the NICCLI Configuration Utility
- Windows RoCE Configuration
- Configuring RSS for Performance in Windows
- Link Aggregation on Ethernet Network Adapters
- Windows Statistics
- Performance Counters

ESXi Configuration Information

Provides information on configuring various Ethernet Adapter features in the ESXi operating system.

The following list provides common ESXi configuration options:

- Supported Operating Systems for Ethernet Network Adapters
- Installing the VMware Driver on Ethernet Network Adapters
- Updating the Firmware Manually on Linux/ESX
- Installing the NICCLI Configuration Utility
- Configuring RDMA over Converged Ethernet (RoCE) on VMware
- Enhanced Network Stack (ENS)
- VMware SR-IOV Use Case Example

TruFlow

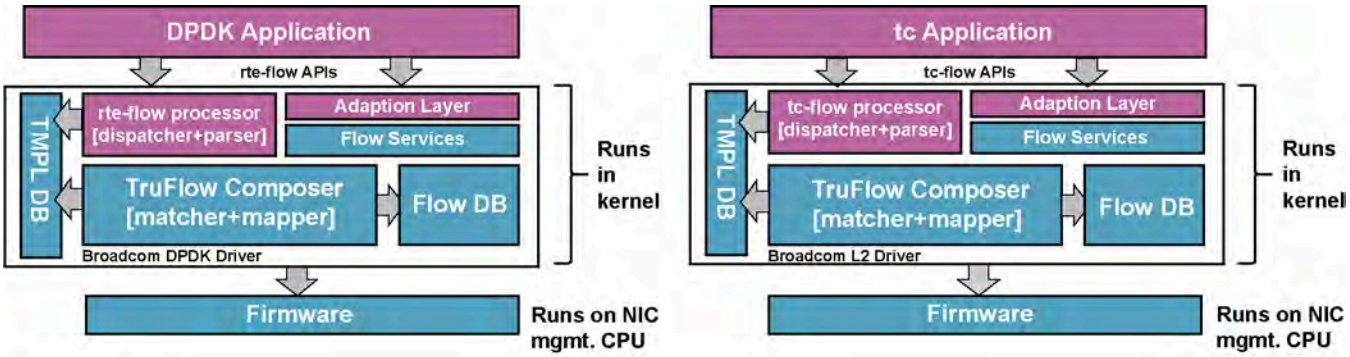
Provides configuration information for TruFlow.

TruFlow™ is the hardware offload of a packet flow classifier on the Broadcom BCM95750X (Thor) and BCM957608 (Thor 2) Ethernet adapters. This allows control of packets based on flows determined by matching well-known packet fields and metadata. TruFlow relies on the CFA (Configurable Flow Accelerator) hardware block in the ASIC which provides the TruFlow features.

The flow classification can be described either by OpenFlow implemented in the Open vSwitch package or by the TC Flower implemented in the iproute2 package. Offload of the flow classifier provides a mechanism to both increase throughput and reduce CPU utilization for users of flow-based systems.

This document provides instructions to enable and configure TruFlow. The instructions for enabling the TruFlow engine in the ASIC are provided in Linux through the TC Flower interface, DPDK's `rte_flow` API, or a custom interface. In this document, TC Flower is used to demonstrate TruFlow with minimal overhead.

Figure 41: TruFlow Block Diagram



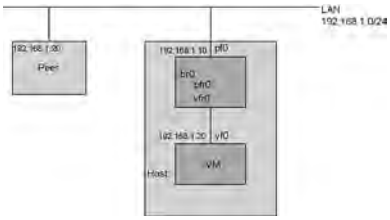
TruFlow consists of the following sections:

- System Configuration
- Enabling TruFlow
- Configuring the Network on the Host
- Creating a VM
- Configuring Networking on the VM
- Testing TruFlow
- Sample Logs
- Supported Patterns and Actions
- Conclusion

System Configuration

Provides information on a basic system configuration for TruFlow.

Figure 42: System Configuration



- The system configuration consists of two PCs: Peer and Host.
- The Host is equipped with a Broadcom BCM95750X (Thor) or BCM957608 (Thor2) Ethernet network adapter with the latest OOB driver, which is configured to demonstrate TruFlow.
- Both the Peer and Host are connected to a common switch.
- A VM is created on the Host to demonstrate the use of TruFlow to control traffic between the Peer and the VM.
- To add networking capability to the VM, bridge br0 was created, as well as a virtual function VF0 on the physical function PF0 and a representor VFR0.
- PF0 and the representor VFR0 were added to the bridge br0 as well as VF0 to the VM.

All commands in this document must be run with superuser privileges.

Enabling TruFlow

Provides information on enabling TruFlow.

To enable TruFlow, SR-IOV, and to set the global resource strategy to static in the network adapter firmware use the NICCLI utility. In this case, BCM95750X is installed in the PCIe slot 01:00.0 and the PF0 is enp1s0npf0.

Note: All bold portions must be updated according to the settings of the test system.

Enabling TruFlow on the Host

To enable TruFlow on the host:

```
niccli -i <index> nvm --setoption enable_truflow --scope 0 --value 1
# Enable truflow on port1 (second port). Cmd not needed for single-pf boards
niccli -i <index> nvm --setoption enable_truflow --scope 1 --value 1
niccli -i <index> nvm --setoption enable_sriov --value 1
The following command is valid only for Thor and should not be used in Thor2 configurations
niccli -i <index> nvm --setoption afm_rm_resc_strategy --value 1
reboot
```

Expected behavior:

- Enumerated BCM95750X/BCM957608 device is seen using lspci.
- All NICCLI commands are executed successfully.

Configuring the Network on the Host

Provides information on configuring the network on the host.

The networking environment is configured as depicted in System Configuration. This section is divided into the following major steps:

- Setting Up the VF
- Setting Up the Representor
- Setting Up the Bridge

Setting Up the VF

To enable trusted virtual function VF0 on PF0 (enp1s0npf0) on the host:

Enabling the VF on the Host

Note: All bold portions must be updated according to the settings of the test system.

To set up the VF on the host:

```
PF_iface=enp1s0npf0
ip link set $PF_iface up
echo 1 > /sys/class/net/$PF_iface/device/sriov_numvfs

# the following command is required only when used with DPDK

ip link set ens1f0np0 vf 0 trust on
```

Expected behavior:

- Link on the PF interface is up
- VF0 with “trust on” is set on PF0 (example: enp1s0npf0) and a new interface is set for VF0 (example: enp2s0v0).

Note: In this case, VF0 is enumerated as device 02:00.0.

Disabling the VF on the Host

To disable the VF on the host:

```
echo 0 > /sys/class/net/$PF_iface/device/sriov_numvfs
```

Setting Up the Representor

To add a representor VFR0, use the devlink utility. In this case, the created representor interface is eth0:

Adding a Representor on the Host

To add a representor on the host:

```
# PF's domain/bus/device/function ID
PF_bdf=0000:01:00.0
devlink dev eswitch set pci/$PF_bdf mode switchdev
```

Expected behavior:

- “devlink dev eswitch show pci/$PF_bdf” should show mode switchdev
- “ip link” should show a new VFR0 (eth0) interface

Removing a Representor from the Host

To remove a representor from the host:

```
devlink dev eswitch set pci/$PF_bdf mode legacy
devlink dev eswitch show pci/$PF_bdf
```

Setting Up the Bridge

Create a bridge br0 and add PF0 (enp1s0npf0) with representor VFR0(eth0):

Creating a Bridge on the Host

To create a bridge on the host:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
BR_IP=192.168.1.10
ip link add br0 type bridge
ip addr flush dev $PF_iface
ip addr add $BR_IP/24 dev br0
ip link set $PF_iface master br0
ip link set $VFR_iface master br0
ip link set br0 up
```

Expected behavior:

- Bridge br0 is up and working.
- At this point, ping the bridge IP on the host (192.168.1.10) from the Peer and vice versa.

Removing a Bridge from the Host

To remove a bridge from the host:

```
ip link set dev br0 type bridge stp_state 0
ip link set $VFR_iface nomaster
ip link set $PF_iface nomaster
ip link del br0
```

Creating a VM

Provides instructions to install Ubuntu VM inside CentOS8 Host.

The following example relies on the virt-manager's supporting tools. However, the same exercise can be accomplished using other virtual machine frameworks such as Oracle VirtualBox or VMware ESX.

Creating a VM on the Host

To create a VM on the host:

```
sed -i 's/SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
reboot
dnf groupinstall "Virtualization Host"
dnf install virt-install
systemctl start libvirtd
mkdir -pv /kvm/{disk,iso}
cd /kvm/iso/
wget http://releases.ubuntu.com/22.04/ubuntu-22.04.1-desktop-amd64.iso

# Note the virt-install command spans multiple lines in this document
# It should be a single command on the terminal
```

```
virt-install --name udesktop22_04-01 --os-type linux --os-variant ubuntu22.04 --ram 4096 --disk /kvm/disk/
udesktop22_04-01.img,device=disk,bus=virtio,size=20,format=qcow2 --graphics vnc,listen=0.0.0.0 --noautoconsole
 --hvm --cdrom /kvm/iso/ubuntu-22.04.1-desktop-amd64.iso --boot cdrom,hd

virsh start udesktop22_04-01

virsh vncdisplay udesktop22_04-01

# At this point the user can launch VNC viewer from an external PC,
# and connect to the VNC server on the virtual machine that we created.

# Complete the OS setup through the VNC client
```

Removing a VM from the Host

To remove a VM from the host:

```
virsh destroy udesktop22_04-01
virsh undefine --remove-all-storage udesktop22_04-01
rm -rf /kvm/
```

Note: The standard VM installation comes with default NAT interfaces virbr0 and vnet0 on subnet 192.168.122/24, which is not used. Use subnet 192.168.1/24 for testing.

Configuring Networking on the VM

Provides information on configuring networking on the VM.

To configure networking on the VM, add VF0 to the VM. It is on PCIe slot 02:00.0, which was described in vf0.xml. This file needs must be created. Note the address domain portion of the XML file must match the domain/bus/device/function portion of VF0.

Creating the vf0.xml File

Create file vf0.xml with the following content:

```
<hostdev mode='subsystem' type='pci' managed='yes'>
 <driver name='vfio'/>
 <source>
    <address domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
 </source>
</hostdev>
```

Adding VF0 to the VM

Use the virsh utility on the Host to add VF0 to the VM from the created file vf0.xml.

```
virsh shutdown udesktop22_04-01
virsh attach-device udesktop22_04-01 vf0.xml --config
virsh start udesktop22_04-01
```

Sample log output on the VM terminal:

```
brcm@brcm-KVM:~$ lshw -businfo -class network

WARNING: you should run this program as super-user.
```

```
Bus info          Device      Class          Description
=========================================================
pci@0000:01:00.0              network        Virtio network device
virtio@0          enp1s0      network        Ethernet interface
pci@0000:07:00.0  enp7s0      network        BCM5750X NetXtreme-E Ethernet Virtual Function
```

- The VF attached to the VM should be enumerated as a separate interface. In the previous example, this is enp7s0 enumerated as 0000:07:00.0
- The Virtio network interface (enp1s0 in the previous example) is not used. Use the interface that is associated with the VF.
- The IP addresses can be successfully assigned to this VF interface and can ping the Peer (and the Peer can ping the VF interface on the VM).

Testing TruFlow

Provides information on testing TruFlow.

In this section, TruFlow is tested using the TC flower utility.

VM Ingress:

- A filter is demonstrated that rewrites the source address of ping echo request packets.
- A filter is also demonstrated which drops all IP traffic to the VM.

VM Egress:

- A filter is demonstrated that rewrites source address of ping echo reply packets.
- A filter is also demonstrated which drops all IP traffic to the Peer.

To begin the test:

1. Start a ping from the Peer to the VM.
2. Run tcpdump on the VM.
3. Continue with the following sections.

Host (VM Ingress)

Note: All bold portions must be updated according to the settings of the test system.

This test (VM ingress) is executed as follows:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
VM_IP=192.168.1.20
PEER_IP=192.168.1.30

# add device to the queue
tc qdisc add dev $PF_iface ingress

# add a filter to rewrite source IP address
tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action pedit ex munge ip src
 set 33.33.33.33 pipe action mirred egress redirect dev $VFR_iface

# show the filter
tc -s filter show dev $PF_iface ingress
```

```
# remove the filter
tc filter del dev $PF_iface ingress

# add a filter to drop traffic
tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action drop

# show the filter
tc -s filter show dev $PF_iface ingress

# remove the filter
tc filter del dev $PF_iface ingress

# remove device from the queue
tc qdisc del dev $PF_iface ingress
```

Host (VM Egress)

The test (VM egress) is executed as follows:

```
PF_iface=enp1s0npf0
VFR_iface=eth0
VM_IP=192.168.1.20
PEER_IP=192.168.1.30

# add device to the queue
tc qdisc add dev $VFR_iface ingress

# add a filter to rewrite source IP address
tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action pedit ex munge ip
 src set 44.44.44.44 pipe action mirred egress redirect dev $PF_iface

# show the filter
tc -s filter show dev $VFR_iface ingress

# remove the filter
tc filter del dev $VFR_iface ingress

# add a filter to drop traffic
tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action drop

# show the filter
tc filter show dev $VFR_iface ingress

# remove the filter
tc filter del dev $VFR_iface ingress

# remove device from the queue
tc qdisc del dev $VFR_iface ingress
```

Note: The `skip_sw` argument ensures that the flow is executed in hardware.

Expected behavior:

- The `in_hw` field in the output of `tc filter show dev $PF_iface` ingress indicates that the flow is in the hardware.
- When the filter is added, the tcpdump and the ping outputs differ based on the applied filter.
- When the source IP is changed with the ingress filter, the source IP changes for ICMP echo requests in the tcpdump on the VM.
- When the source IP is changed with the egress filter, the source IP changes for ICMP echo reply in ping output on the Peer.
- When traffic is dropped, the ping output stops and tcpdump stops showing ICMP packets.

Sample Logs

Provides sample logs from the test.

Host Log – Network Configuration

The host log of the setup is as follows:

```
# ifconfig enp1s0npf0 up
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen
 1000
link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff

# echo 1 > /sys/class/net/enp1s0npf0/device/sriov_numvfs
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen
 1000
link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto, trust off
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff

ip link set $PF vf 0 trust on

Command Executed Successfully.

# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
```

```
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen
 1000
 link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto, trust on
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff

# devlink dev show

pci/0000:01:00.0
pci/0000:02:00.0

# devlink dev eswitch set pci/0000:01:00.0 mode switchdev
# devlink dev eswitch show pci/0000:01:00.0

pci/0000:01:00.0: mode switchdev

# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen
 1000
link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto, trust on
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff
6: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN mode DEFAULT group default
 qlen 1000
link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff

# ip link add br0 type bridge
# ip addr add 192.168.1.10/24 dev br0
# ip link set eth0 master br0
# ip link set enp1s0npf0 master br0
# ip link set br0 up
# ip l

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 58:11:22:4b:d1:c7 brd ff:ff:ff:ff:ff:ff
4: enp1s0npf0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state UP mode DEFAULT group de-
fault qlen 1000
link/ether 00:62:0b:fa:5c:60 brd ff:ff:ff:ff:ff:ff
vf 0     link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff, spoof checking off, link-state auto, trust on
5: enp2s0v0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
link/ether 8e:37:99:15:8c:f0 brd ff:ff:ff:ff:ff:ff
6: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UNKNOWN mode DEFAULT group
 default qlen 1000
link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff
```

```
>7: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen
 1000
          link/ether 00:62:0b:6d:12:5e brd ff:ff:ff:ff:ff:ff
```

Host Log – Testing TruFlow

The host log of the test is as follows:

```
# PF_iface=enp1s0f0np0
# VFR_iface=eth0
# VM_IP=192.168.1.20
# PEER_IP=192.168.1.30
# tc qdisc add dev $PF_iface ingress
# tc qdisc show dev $PF_iface ingress
qdisc ingress ffff: parent ffff:fff1 ----------------
# tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action pedit ex munge ip
 src set 33.33.33.33 pipe action mirred egress redirect dev $VFR_iface
# tc filter show dev $PF_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.20
  skip_sw
  in_hw in_hw_count 1
    action order 1:  pedit action pipe keys 1
      index 1 ref 1 bind 1
     key #0  at ipv4+12: val 21212121 mask 00000000
    used_hw_stats delayed

    action order 2: mirred (Egress Redirect to device eth0) stolen
    index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $PF_iface ingress
# tc filter add dev $PF_iface protocol ip parent ffff: flower skip_sw dst_ip $VM_IP action drop
# tc filter show dev $PF_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.20
  skip_sw
  in_hw in_hw_count 1
    action order 1: gact action drop
     random type none pass val 0
     index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $PF_iface ingress
# tc qdisc del dev $PF_iface ingress
# tc qdisc add dev $VFR_iface ingress
# tc qdisc show dev $VFR_iface ingress
qdisc ingress ffff: parent ffff:fff1 ----------------
# tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action pedit ex munge
 ip src set 44.44.44.44 pipe action mirred egress redirect dev $PF_iface
# tc filter show dev $VFR_iface ingress
```

```
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.30
  skip_sw
  in_hw in_hw_count 1
    action order 1:  pedit action pipe keys 1
      index 1 ref 1 bind 1
     key #0  at ipv4+12: val 2c2c2c2c mask 00000000
    used_hw_stats delayed

    action order 2: mirred (Egress Redirect to device enp9s0f0np0) stolen
    index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $VFR_iface ingress
# tc filter add dev $VFR_iface protocol ip parent ffff: flower skip_sw dst_ip $PEER_IP action drop
# tc filter show dev $VFR_iface ingress
filter parent ffff: protocol ip pref 49152 flower chain 0
filter parent ffff: protocol ip pref 49152 flower chain 0 handle 0x1
  eth_type ipv4
  dst_ip 192.168.1.30
  skip_sw
  in_hw in_hw_count 1
    action order 1: gact action drop
     random type none pass val 0
     index 1 ref 1 bind 1
    used_hw_stats delayed
# tc filter del dev $VFR_iface ingress
# tc qdisc del dev $VFR_iface ingress
```

VM Log

The VM log is as follows:

```
brcm@brcm-KVM-22:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
        link/ether d0:94:66:42:61:59 brd ff:ff:ff:ff:ff:ff
        altname enp1s0f0
        inet 10.136.14.59/24 brd 10.136.14.255 scope global eno1
          valid_lft forever preferred_lft forever
        inet6 fe80::d294:66ff:fe42:6159/64 scope link
          valid_lft forever preferred_lft forever
4: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
        link/ether d0:94:66:42:61:5a brd ff:ff:ff:ff:ff:ff
        altname enp1s0f1
39: enp2s0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 5000 qdisc mq state UP group default qlen 1000
        link/ether 00:0a:f7:31:43:f0 brd ff:ff:ff:ff:ff:ff
```

```
        inet 192.168.1.30/24 brd 192.168.1.255 scope global enp2s0np0
          valid_lft forever preferred_lft forever
        inet6 fe80::20a:f7ff:fe31:43f0/64 scope link
          valid_lft forever preferred_lft forever
brcm@brcm-KVM-22:~$ sudo tcpdump -ni enp7s0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp7s0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:13:25.919610 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:25.919622 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:26.377695 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 8, length 64
14:13:26.377867 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 8, length 64
14:13:27.402244 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 9, length 64
14:13:27.402418 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 9, length 64

... (above is a nornal traffic showing ICMP and STP packets)

14:13:33.962551 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:33.962561 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:34.473590 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 16, length 64
14:13:35.497581 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 17, length 64
14:13:35.970049 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:35.970070 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:13:36.521562 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 18, length 64
14:13:37.546220 IP 33.33.33.33 > 192.168.1.20: ICMP echo request, id 15, seq 19, length 64

... (above is a traffic after with ingress filter rewriting the ping request source address)

14:13:59.050013 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 40, length 64
14:13:59.050178 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 40, length 64
14:14:00.073636 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 41, length 64
14:14:00.073794 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 41, length 64
14:14:00.109307 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:00.109324 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36

... (above is a normal traffic after the filter was removed)

14:14:18.346006 ARP, Request who-has 192.168.1.20 tell 192.168.1.30, length 46
14:14:18.346096 ARP, Reply 192.168.1.20 is-at 3a:0a:cb:72:1b:a6, length 28
14:14:20.219290 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:20.219309 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:22.228215 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
```

```
14:14:22.228236 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:24.247132 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:24.247155 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:26.265764 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:26.265777 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36

... (above is a traffic when all IP packets to VM were dropped)

14:14:44.969832 ARP, Request who-has 192.168.1.20 tell 192.168.1.30, length 46
14:14:44.969841 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 85, length 64
14:14:44.969934 ARP, Reply 192.168.1.20 is-at 3a:0a:cb:72:1b:a6, length 28
14:14:44.970105 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 85, length 64
14:14:45.993826 IP 192.168.1.30 > 192.168.1.20: ICMP echo request, id 15, seq 86, length 64
14:14:45.993984 IP 192.168.1.20 > 192.168.1.30: ICMP echo reply, id 15, seq 86, length 64
14:14:46.380958 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36
14:14:46.380978 STP 802.1w, Rapid STP, Flags [Proposal, Learn, Forward, Agreement], bridge-id 8001.8c:47:be:d-
f:82:80.8100, length 36

...(above is a normal traffic after the filter was removed)
...
^C
517 packets captured
517 packets received by filter
0 packets dropped by kernel
brcm@brcm-KVM-22:~$
```

Peer Log

The peer log is as follows:

```
brcm@peer:~$ ip a1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever2:
eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
        link/ether d0:94:66:42:61:59 brd ff:ff:ff:ff:ff:ff
        altname enp1s0f0
        inet 10.136.14.59/24 brd 10.136.14.255 scope global eno1
          valid_lft forever preferred_lft forever
        inet6 fe80::d294:66ff:fe42:6159/64 scope link
          valid_lft forever preferred_lft forever4:
eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
        link/ether d0:94:66:42:61:5a brd ff:ff:ff:ff:ff:ff
```

```
        altname enp1s0f1
39: enp2s0np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 5000 qdisc mq state UP group default qlen 1000
        link/ether 00:0a:f7:31:43:f0 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.30/24 brd 192.168.1.255 scope global enp2s0np0
          valid_lft forever preferred_lft forever
        inet6 fe80::20a:f7ff:fe31:43f0/64 scope link
          valid_lft forever preferred_lft forever
        brcm@peer:~$ ping 192.168.1.20
      PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=0.636 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=0.393 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=1.08 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=0.397 ms
64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=0.397 ms
64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=0.431 ms
64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=0.656 ms
64 bytes from 192.168.1.20: icmp_seq=8 ttl=64 time=0.540 ms
64 bytes from 192.168.1.20: icmp_seq=9 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=10 ttl=64 time=0.960 ms
64 bytes from 192.168.1.20: icmp_seq=11 ttl=64 time=0.685 ms
64 bytes from 192.168.1.20: icmp_seq=12 ttl=64 time=0.463 ms
64 bytes from 192.168.1.20: icmp_seq=13 ttl=64 time=1.06 ms
64 bytes from 192.168.1.20: icmp_seq=14 ttl=64 time=0.463 ms
64 bytes from 192.168.1.20: icmp_seq=15 ttl=64 time=0.662 ms

(there is a gap in the initial icmp_seq when packets are dropped with ingress filter)

64 bytes from 192.168.1.20: icmp_seq=40 ttl=64 time=0.850 ms
64 bytes from 192.168.1.20: icmp_seq=41 ttl=64 time=0.474 ms
64 bytes from 192.168.1.20: icmp_seq=42 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=43 ttl=64 time=0.990 ms
64 bytes from 192.168.1.20: icmp_seq=44 ttl=64 time=0.459 ms
64 bytes from 192.168.1.20: icmp_seq=45 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=46 ttl=64 time=0.902 ms

... (normal ping continues)

64 bytes from 192.168.1.20: icmp_seq=126 ttl=64 time=1.17 ms
64 bytes from 192.168.1.20: icmp_seq=127 ttl=64 time=0.861 ms
64 bytes from 192.168.1.20: icmp_seq=128 ttl=64 time=0.466 ms
64 bytes from 44.44.44.44: icmp_seq=129 ttl=64 time=0.453 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=130 ttl=64 time=0.513 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=131 ttl=64 time=0.657 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=132 ttl=64 time=0.522 ms (DIFFERENT ADDRESS!)

... (the source address of ping reply packets was changed with egress filter)

64 bytes from 44.44.44.44: icmp_seq=150 ttl=64 time=0.399 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=151 ttl=64 time=0.463 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=152 ttl=64 time=0.447 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=153 ttl=64 time=0.650 ms (DIFFERENT ADDRESS!)
64 bytes from 44.44.44.44: icmp_seq=154 ttl=64 time=0.458 ms (DIFFERENT ADDRESS!)
64 bytes from 192.168.1.20: icmp_seq=155 ttl=64 time=1.16 ms
```

```
64 bytes from 192.168.1.20: icmp_seq=156 ttl=64 time=1.02 ms
64 bytes from 192.168.1.20: icmp_seq=157 ttl=64 time=0.961 ms
64 bytes from 192.168.1.20: icmp_seq=158 ttl=64 time=0.443 ms

... (ping replies are back to normal after egress filter was removed)
...
^C
--- 192.168.1.20 ping statistics ---
197 packets transmitted, 142 received, 27.9188% packet loss, time 199643ms
rtt min/avg/max/mdev = 0.388/0.684/1.356/0.272 ms
brcm@peer:~$
```

Supported Patterns and Actions

Provides supported patterns and actions.

The following Match patterns and actions are supported by TruFlow. Any combination of action and match pattern can be used to specify the flow:

Table 81: List of Supported Actions

Actions Supported	Description
Count	Packet count and byte count
Redirect	Forward to VF or PF
RSS	Forward to a set of queues
Queue	Forward to a specific queue
Drop	Drop the packet
VLAN tag push/pop	Add VLAN tag or remove VLAN tag
Tunnel encap	Add Tunnel header like VXLAN
Tunnel decap	Remove Tunnel header like VXLAN
Decrement TTL	Decrement IP TTL by 1
Rewrite header	Modify any of L2 SMAC/DMAC, L3 SRC/DST , L4 SPORT/ DPORT
Resubmit (goto)	Jump to another flow group
Group (chain)	Define Flow group
Priority	Wildcard Priority
Mirror	Mirror the traffic to a specific port
Meter	Flow based meter

Table 82: List of Supported Match Fields

Supported Headers	Supported Fields
Ethernet	Destination MAC, Source MAC, and Ethernet type
VLAN	VLAN ID
IPv4	Source IP, Destination IP, TTL, TOS, Protocol ID

Supported Headers	Supported Fields
IPv6	Source IP, Destination IP, TTL, TC, Protocol ID
TCP/UDP	Source Port, Destination port
VXLAN	VNI

Conclusion

Provides the differences between TC Flower and OpenFlow.

Instructions for enabling TruFlow using TC Flower from iproute2 have been provided. It is also possible to describe TruFlow using OpenFlow from the Open vSwitch package. The following example shows the differences between commands using TC Flower and OpenFlow:

```
# Exact Match

TC: tc filter add dev <vfr0> protocol ip parent ffff: prio 10 chain 0 flower skip_sw src_mac 00:01:22:33:81:60
 dst_mac 00:0a:f7:a6:81:60 action mirred egress redirect dev <pf0>

OF: ovs-ofctl add-flow ovsbr0 ""in_port=vfrep1 dl_dst=00:0a:f7:98:38:60 dl_src=c2:a1:ed:fe:a5:90
 dl_type=0x0800 ipv4 actions=output:trustedvf""

# Wild Card

TC: tc filter add dev <vfr0> protocol ip parent ffff: prio 10 chain 0 flower skip_sw src_mac
 00:01:22:33:81:60/ff:ff:ff:ff:ff:00 dst_mac 00:0a:f7:a6:81:60/ff:ff:ff:ff:ff:00 action mirred egress redirect
 dev <pf0>

OF: ovs-ofctl add-flow ovsbr0 ""in_port=vfrep1 dl_dst=00:0a:f7:98:38:60/ff:ff:ff:ff:ff:00 dl_sr-
c=c2:a1:ed:fe:a5:90/ff:ff:ff:ff:ff:00 dl_type=0x0800 ipv4 actions=output:trustedvf""
```

OVS and TC and OVS-DPDK Offload

Provides information on enabling Open vSwitch (OVS) for hardware offloading on BCM95750X and BCM957608 devices.

Note: This feature is only available with software version 235 and with BCM95750X/BCM957608 devices.

This section consists of the following:

- OVS-TC and OVS-DPDK Feature List
- NVM Changes
- Installing and Using the OVS-TC and OVS-DPDK Application
- Configuring the OVS-TC and OVS-DPDK Application
- OVS Flow Offloads
- Test Guidance

OVS-TC and OVS-DPDK Feature List

Provides the feature list for OVS-TC and OVS-DPDK.

The features listed are based on the 235 software release.

OVS-TC Feature List

Table 83: Supported Software

Feature	Description	Supported
OVS	Upstream Open vSwitch Version Supported.	• OVS 3.3.2 (with 23.11) • OVS 3.5 (with 24.11)
DPDK	DPDK Versions Supported.	• DPDK 23.11 • DPDK 24.11

Table 84: Offload Type

Feature	Description	Supported
Exact Match	Uses hardware-based on-chip exact match lookups.	IPv4 and IPv6
Wildcard Match	Uses hardware-based on-chip wildcard lookups.	IPv4 and IPv6
External Exact Match	Uses host DDR for exact match lookups.	No (hardware does not support)
VF Representor/Scale	Linux kernel switchdev based VF representors.	• 1 port NIC – 128 • 2 port NIC – 128 • 4 port NIC – 128
PF representor/Scale	PF representors.	Not supported
VF to VF Offload	Full offload(VM to VM Local Host)	–

Table 85: Match Fields

Feature	Description	Supported
DMAC	Destination MAC	Yes
SMAC	Source MAC	Yes
VLAN	VLAN tag	Yes
ETH	Ethertype	Yes
SIP	Source IP	Yes
DIP	Destination IP	Yes
ToS	Type of Service/DSCP	Yes
Proto	Protocol	Yes
SPORT	Source Port	Yes
DPORT	Destination Port	Yes
tDMAC	Tunnel Destination MAC	Yes
tSMAC	Tunnel Source MAC	Yes
tVLAN	Tunnel VLAN	No
tETH	Tunnel Ethertype	Yes
tDSCP	Tunnel DSCP	No
tSIP	Tunnel Source IP	Yes
tDIP	Tunnel Destination IP	Yes
tProto	Tunnel Protocol	Yes
tSPORT	Tunnel Source Port	Yes
tDPORT	Tunnel Destination Port	Yes
Tunnel ID	Tunnel identifier	Yes

Table 86: Actions

Feature	Description	Supported
count	Packet and Byte flow statistics.	Yes
redirect	Forward the packets from wire to the VF or VF to the wire.	Yes
queue	Forward the packets to a queue.	No
drop	Drop the packets.	Yes
CT (connection tracking)	Hardware connection tracking.	No
VLAN tag push/pop	Modify VLAN PCP and VID.	Yes
Tunnel encap	Push tunnel header on TX.	VxLAN
Tunnel decap	Pop tunnel header on RX.	VxLAN
decrement ttl	–	No

Feature	Description	Supported
rewrite header	Modify header fields, please note some combinations might be restricted.	SMAC, DMAC, SIPv4, DIPv4, SPORT, DPORT
resubmit (goto)	–	Yes
group (chain)	–	No
priority	Wildcard flows only can handle priority, EM are considered high priority than all WC flows.	Yes
mark	Mark support.	No

Table 87: Match Pattern

Feature	Description	Supported
Ingress WC: DMAC/SMAC/VLAN/DIP/SIP/ Proto/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/2048 Flows • IPv6 Only/1024 Flows
Ingress EM: DMAC/SMAC/VLAN/Proto / DIP/SIP/DPORT/SPORT	EM – Exact Match lookup, non Masked	• IPv4 Only = 5460 • IPv6 Only = 3276
Ingress WC: tDMAC/tSMAC/tVLAN/tDIP/ tSIP/tProto/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/2048 Flows • IPv6 Only/1024 Flows
Ingress EM: tDMAC(VxLAN)/Tunnel ID/ DMAC/SMAC/Proto/DIP/SIP/DPORT/ SPORT	EM – Exact Match lookup, non Masked	• IPv4-IPv4=4095 • IPv4-IPv6=2730
Ingress WC: tDMAC(VxLAN) /Tunnel ID/ DMAC/SMAC/Proto/DIP/SIP/DPORT/ SPORT	WC – Wildcard lookup with Mask	• IPv4-IPv4=1024 • IPv4-IPv6=1024
Egress WC: DMAC/SMAC/VLAN/Proto/ DIP/SIP/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/2048 Flows • IPv6 Only/1024 Flows
Egress EM: DMAC/SMAC/VLAN/Proto/ DIP /SIP/DPORT/SPORT	EM – Exact Match lookup, non Masked	• IPv4 Only = 5461 • IPv6 Only = 3276
Egress WC: DMAC/SMAC/VLAN/Proto/ DIP/SIP/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only = 2048 Flows • IPv6 Only = 1024 Flows
Ingress WC: DMAC/SMAC[Incremental DMAC Case]	WC – Wildcard lookup with Mask	IPv4/270 Flows
Ingress EM: DMAC(Mask) /SMAC(Only EM) [Incremental DMAC Case]	EM – Exact Match lookup, non Masked SMAC	IPv6 / 270 Flows
Egress WC: DMAC/SMAC [Incremental DMAC and SMAC Case]	WC – Wildcard lookup with Mask	• IPv4/2048 Flows • IPv6/1024 Flows
Egress EM: DMAC/SMAC [Incremental DMAC and SMAC Case]	EM – Exact Match Lookup, Both DMAC and SMAC are non Masked	• IPv4/8192 Flows • IPv6/8192 Flows

Table 88: Other Features

Feature	Description	Supported
Multiple Cards	Multiple NIC cards on the same host.	Yes
Multi-Root	NUMA specific flow offloads.	No
Multiple Instances	Ability to start multiple vSwitch applications on the same host.	No
Error Recovery	Firmware reset and recovery.	No, Requires restarting the application
Multi-Host	Ability to start multiple vSwitch applications on different hosts.	No
High Availability/Hot Upgrade	Allow In-Service Software Upgrades.	No, Requires restarting the application

OVS-DPDK Feature List

Table 89: Supported Software

Feature	Description	Supported
OVS	Upstream Open vSwitch Version Supported.	• OVS 3.3.2 (with 23.11) • OVS 3.5 (with 24.11)
DPDK	DPDK Versions Supported.	• DPDK 23.11 • DPDK 24.11

Table 90: Offload Type

Feature	Description	Supported
Exact Match	Uses hardware-based on-chip exact match lookups.	IPv4/IPv6
Wildcard Match	Uses hardware-based on-chip wildcard lookups.	IPv4/IPv6
External Exact Match	Uses host DDR for exact match lookups.	No (hardware does not support)
VF Representor/Scale	DPDK based VF representors [1 VF per port is used for uplink from the port, for remaining VFs representor functions are created].	• 1 port NIC – 127 • 2 port NIC – 126 • 4 port NIC – 124
PF representor/Scale	PF representors.	Not supported.
VF to VF Offload	Full offload (VM to VM Local Host).	Yes

Table 91: Match Fields

Feature	Description	Supported
DMAC	Destination MAC	Yes
SMAC	Source MAC	Yes
VLAN	VLAN tag	Yes

Feature	Description	Supported
ETH	Ethertype	Yes
SIP	Source IP	Yes
DIP	Destination IP	Yes
ToS	Type of Service/DSCP	No
Proto	Protocol	Yes
SPORT	Source Port	Yes
DPORT	Destination Port	Yes
tDMAC	Tunnel Destination MAC	Yes
tSMAC	Tunnel Source MAC	Yes
tVLAN	Tunnel VLAN	Yes
tETH	Tunnel Ethertype	Yes
tDSCP	Tunnel DSCP	No
tSIP	Tunnel Source IP	Yes
tDIP	Tunnel Destination IP	Yes
tProto	Tunnel Protocol	Yes
tSPORT	Tunnel Source Port	Yes
tDPORT	Tunnel Destination Port	Yes
Tunnel ID	Tunnel identifier	Yes

Table 92: Actions

Feature	Description	Supported
count	Packet and Byte flow statistics.	Yes
redirect	Forward the packets from wire to the VF or VF to the wire.	Yes
queue	Forward the packets to a queue.	No
drop	Drop the packets.	Yes
ct (connection tracking)	Hardware connection tracking.	No
VLAN tag push/pop	Modify VLAN PCP and VID.	Yes
Tunnel encap	Push tunnel header on TX.	VxLAN
Tunnel decap	Pop tunnel header on RX.	VxLAN
decrement ttl	–	Yes
rewrite header	Modify header fields, please note some combinations might be restricted.	SMAC, DMAC, SIPv4, DIPv4, SPORT, DPORT
resubmit (goto)	–	No
group (chain)	–	No
priority	Wildcard flows only can handle priority, EM are considered high priority than all WC flows.	No

Feature	Description	Supported
mark	Mark support.	No

Table 93: Match Patterns

Feature	Description	Supported
Ingress WC: DMAC/SMAC/VLAN/DIP/SIP / Proto/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/2048 Flows • IPv6 Only/1024 Flows
Ingress EM: DMAC/SMAC/VLAN/Proto / DIP/SIP/DPORT/SPORT	EM – Exact Match lookup, non Masked	• IPv4 Only = 5460 • IPv6 Only = 3276
Ingress WC: tDMAC/tSMAC/tDIP/tSIP/ tProto/DPORT/SPORT	WC – Wildcard lookup with Mask	IPv4 Only/1024 Flows
Ingress EM: tDMAC(VxLAN)/Tunnel ID/ DMAC/SMAC/Proto/DIP/SIP/DPORT/ SPORT	EM – Exact Match lookup, non Masked	• IPv4 Only = 5460 • IPv6 Only = 3276
Ingress WC: tDMAC(VxLAN) /Tunnel ID/ DMAC/SMAC/Proto/DIP/SIP/DPORT/ SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/1024 Flows • IPv6 Only/1024 Flows
Egress WC: DMAC/SMAC/VLAN/Proto/ DIP/SIP/DPORT/SPORT	WC – Wildcard lookup with Mask	• IPv4 Only/2048 Flows • IPv6 Only/1024 Flows
Egress EM: DMAC/SMAC/VLAN/Proto/ DIP /SIP/DPORT/SPORT	EM – Exact Match lookup, non Masked	• IPv4 Only/16384/3 = 5461 • IPv6 Only/16384/5 = 3276
Ingress WC: DMAC/SMAC[Incremental DMAC Case]	WC – Wildcard lookup with Mask	IPv4/270 Flows (Must include Ethertype)
Ingress EM: DMAC (Mask) /SMAC (Only EM) [Incremental DMAC Case]	EM – Exact Match lookup, non Masked SMAC	IPv6/270 Flows (Must include Ethertype)
Egress WC: DMAC/SMAC [Incremental DMAC and SMAC Case]	WC – Wildcard lookup with Mask	• IPv4/2048 Flows • IPv6/1024 Flows
Egress EM: DMAC/SMAC [Incremental DMAC and SMAC Case]	EM – Exact Match Lookup, Both DMAC and SMAC are non Masked	• IPv4/2048 Flows • IPv6/1024 Flows • (For this key it chooses WC now) • (Must include Ethertype)

Table 94: Other Features

Feature	Description	Supported
Multipe Cards	Multiple NIC cards on the same host.	Yes
Multi-Root	Multiple NIC cards on the same host.	No
Multiple Instances	Ability to start multiple vSwitch applications on the same host.	No
Error Recovery	Firmware reset and recovery.	No, Requires restarting the application
Multi-Host	Ability to start multiple vSwitch applications on different hosts.	No
High Availability/Hot Upgrade	Allow In-Service Software Upgrades.	No, Requires restarting the application

NVM Changes

Provides information on updating the NVM settings for the OVS and TC applications.

To update the NVM settings:

1. Restore the factory defaults using the following comamnd:
   ```
   niccli --dev ${PF0} nvm --restore_factory_defaults
   ```
2. Enable the NVM settings shown in the following table.

Table 95: NVM Settings

NVM Option	NVM/Scope	Value	Comments
933	`enable_truflow` /Function	1	Must be set on All PFs.
1101	`afm_rm_resc_strategy` / Device	1	Global setting for AFM resource strategy to Static.
401	`enable_sriov` /Device	1	Enable SR-IOV
408	`number_of_vfs_per_pf` / Function	#VFs	Number of VFs per PF

Note: `afm_rm_resc_strategy` is only supported on BCM95750X devices.

Example:

```
PF0=p1p1
niccli --dev ${PF0} nvm --restore_factory_defaults

# Enable Truflow on PF0
niccli --dev ${PF0} nvm --setoption enable_truflow --scope 0 --value 0x01
# Enable Trflow on PF1
niccli --dev ${PF0} nvm --setoption enable_truflow --scope 1 --value 0x01

#Enable SRIOV
niccli --dev ${PF0} nvm --setoption enable_sriov --value 0x01

#  Increase the number of VFs from default.
niccli --dev ${PF0} nvm --setoption number_of_vfs_per_pf --scope 0 --value 0x0008
niccli --dev ${PF0} nvm --setoption number_of_vfs_per_pf --scope 1 --value 0x0008

niccli --dev ${PF0} nvm --setoption afm_rm_resc_strategy --value 0x01
```

Note: `afm_rm_resc_strategy` is only supported on BCM95750X devices.

3. Reboot or power cycle the device to ensure the system has been reset.

Installing and Using the OVS-TC and OVS-DPDK

Provides information on installing and using the OVS-TC and OVS-DPDK.

Installing and Using the OVS-TC

This section describes how to build and install OVS with `bnxt_en` . To install OVS, use the following commands:

1. Download and install the L2 driver from the latest software CD:
   ```
   #tar -xvf $BCMCD/drivers_linux/bundle/netxtreme-bnxt_en-1.10.3-236.1.6.0.tar.gz
   ```

```
#cd netxtreme-bnxt_en-1.10.3-236.1.6.0
#make && make install
```

2. Download and install the Open vSwitch using the following commands:

```
#tar -xvf $BCMCD/utils/ovs_package/ovs-src-3.5.0-236.1.5.0.tar.gz
#cd ovs-src-3.5.0-236.1.5.0
#TARGET=`uname -r`
./boot.sh && \
./configure --with-linux=/lib/modules/${TARGET}/build
#make -j `nproc`  && sudo make install
```

Note: Use the inbox `openvswitch.ko` module, do NOT build it.

Installing and Using the OVS-DPDK

This section describes how to build and install OvS 3.5 with DPDK 24.11. See the detailed installation and system requirements for Linux as follows:

1. OVS with DPDK – https://docs.openvswitch.org/en/latest/intro/install/dpdk/
2. DPDK – https://doc.dpdk.org/guides/linux_gsg/

Installation Prerequisites (Meson and Ninja)

The latest version of meson and ninja must be installed on the server. The following scripts can be used to retrieve them.

Note: These scripts were tested on Ubuntu 18.04/20.04, but should work on other Linux platforms. Ensure to use python (3.5 or later). Copy into the .sh file and source that file. The `host_install_meson_ninja` can be run from that shell.

```
host_install_meson_ninja()
{
if [ ! -d "/var/lib/tftpboot/MESON-NINJA" ];then
echo -n "If you wish to download meson and ninja build tools: (y/n)"
read resp
if [ x"$resp" == "xy" ]; then
mkdir -p /var/lib/tftpboot/MESON-NINJA
cd /var/lib/tftpboot/MESON-NINJA
echo "Downloading meson..."
wget -O meson.zip https://github.com/mesonbuild/meson/archive/master.zip
echo "Downloading ninja..."
wget -O ninja.zip https://github.com/ninja-build/ninja/archive/master.zip
fi
else
echo "Using MESON-NINJA tools have been downloaded already..."
#cd /var/lib/tftpboot/MESON-NINJA
#ls -l
return 0
fi
echo "Installing meson"
cd /var/lib/tftpboot/MESON-NINJA && unzip -oq meson.zip
ln -sf /var/lib/tftpboot/MESON-NINJA/meson-master/meson.py /usr/local/bin/meson

echo "Installing ninja"
cd /var/lib/tftpboot/MESON-NINJA && unzip -oq ninja.zip
cd /var/lib/tftpboot/MESON-NINJA/ninja-master && python3 configure.py --bootstrap && cp ninja /usr/bin/.
echo "Done installing meson and ninja"}
```

Downloading and Installing OVS and DPDK

To download and install the OVS and DPDK applications:

1. Download the OVS and DPDK source from the latest software CD.
   a. Example:
      - `$BCMCD/utils/ovs_package/ovs-src-3.5.0-236.1.5.0.tar.gz`
      - `$BCMCD/drivers_dpdk/dpdk-24.11.0-236.1.9.0.tar.gz`
   b. Extract the DPDK folder and set the following environment variables:
      ```
      export RTE_SDK=”path/to/your/dpdk/src”
      cd $RTE_SDK
      ```

2. Install DPDK using meson (see details here).
   a. Set the `RTE_TARGET` to the build folder:
      ```
      export RTE_TARGET=x86_64-native-linuxapp-gcc
      ```
   b. Build and install the DPDK library:
      ```
      meson $RTE_TARGET  --buildtype=debug
      ninja -C $RTE_TARGET
      sudo ninja -C $RTE_TARGET install
      sudo ldconfig
      ```

      For scaling tests, if the number of ports exceeds 32, update the meson config param as follows:
      ```
      ovs-vsctl set Open_vSwitch . other_config:per-port-memory=false
      ```
      Note: DPDK_MESON_CFG : -Dmax_ethports=<Number of total Reps + Trusted VFs>
   c. Determine if `libdpdk` can be found by `pkg-config` . The following command returns the DPDK version installed. If it is not found, export the path to the installed DPDK libraries: `export PKG_CONFIG_PATH=/path/to/installed/".pc" file/for/DPDK`
      a. `pkg-config --modversion libdpdk`

         Note: For Centos: `export PKG_CONFIG_PATH=/usr/local/lib64/pkgconfig`
   d. To fix `buildtools/meson.build:45:8: ERROR: Problem encountered: missing python module: elftools` , update using the following commands:
      a. `sudo apt update`
      b. `sudo apt install python3-pyelftools (RHEL/CentOS: pip3 install pyelftools`
   e. To load the library for handling NUMA(Non Uniform Memory Access), use the following commands:
      a. `sudo apt-get install libnuma-dev (RHEL/CentOS: yum install numactl-devel)`
3. Install OVS by extracting the OVS folder and setting the following environment variables:
   ```
   export OVS_DIR=”path/to/your/ovs/src"
   export OVS_CFLAGS='-march=native -g -Ofast'
   cd $OVS_DIR
   ./boot.sh
   ./configure CFLAGS="${OVS_CFLAGS} -DALLOW_EXPERIMENTAL_API"  --with-
   dpdk=static
   make -j `nproc`
   sudo make install
   export PATH=$PATH:$OVS_DIR/utilities
   ```

Configuring OVS-TC and OVS-DPDK

Provides information configuring both OVS-TS and OVS-DPDK.

Configuring OVS-TC

This section provides configuration information for OVS-TC.

Creating VFs

To create the VFs, use the following commands:

1. Load the L2 driver using the following command:
   ```
   modprobe bnxt_en
   ```
2. Bring up the PFs using the following command:
   ```
   ifconfig ${PF0} up
   ```
3. Create the VFs using the following command:
   ```
   echo $num_vfs > /sys/class/net/${PF0}/device/sriov_numvfs
   ```

Creating switchdev Ports on PF0

To create switchdev ports on PF0, use the following commands:

```
ovs_tc_destroy_vfreps_on_pf0()
{
export PF0=`ls -l /sys/class/net | grep ${PCI_PF0} | awk '{print $9}'`
if [ -z "$PF0"  ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: Could not find PF0."
echo "       To fix, please try to rebuild bnxt_en."
return
fi

echo "Destroying existing VF-reps on $PF0"
#sudo devlink dev show
sudo devlink dev eswitch show pci/${PCI_PF0}
sudo devlink dev eswitch set pci/${PCI_PF0} mode legacy
sleep 2
echo ""

}

ovs_tc_create_vfreps_on_pf0()
{
export PF0=`ls -l /sys/class/net | grep ${PCI_PF0} | awk '{print $9}'`
if [ -z "$PF0"  ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: Could not find "
echo "       To fix, please try to rebuild bnxt_en."
return
fi

echo "Creating VF-reps on $PF0"
#sudo devlink dev show
sudo devlink dev eswitch show pci/${PCI_PF0}
```

```
sudo devlink dev eswitch set pci/${PCI_PF0} mode switchdev
sleep 2
echo ""
}
```

Starting and Initializing OVS with TC

To start and initialize OVS with TC, use the following commands:

```
#OVS_HW_OFFLOAD=true
ovs_tc_db_init()
{
echo "Remove the old ovs-vswitchd.log file"
cat /dev/null > /usr/local/var/log/openvswitch/ovs-vswitchd.log

if [ ! -f $OVS_DIR/utilities/ovs-ctl ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: ovs-ctl not built."
echo "       To fix, please try to rebuild OvS."
return
fi

timeout 5 ovs-vsctl --if-exists del-br $OVS_BR0
timeout 5 ovs-vsctl --if-exists del-br $OVS_BR1
timeout 5 ovs-ctl stop

sleep 5
echo "Make sure OvS has stopped..."
ovs-ctl status
sleep 1

/sbin/modprobe act_mirred
echo "Loading openvswitch kernel module..."
/sbin/modprobe openvswitch
sleep 1
/sbin/lsmod | grep openvswitch
if [ $? -eq 1 ]; then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: openvswitch kernel module was not loaded"
echo "       To fix, Check dmesg to see why"
return
fi

echo "First start ovsdb server without ovs-vswitchd"
ovs-ctl --no-ovs-vswitchd start

echo "Setting OVS TC params..."
set -x
ovs-vsctl --no-wait clear Open_vSwitch . other_config


# Enable/Disable HW-Offloads
ovs-vsctl --no-wait set Open_vSwitch . other_config:hw-offload=$OVS_HW_OFFLOAD
sleep 1
```

```
echo ""

#ovs-vsctl --no-wait set Open_vSwitch . other_config:emc-insert-inv-prob=0
# This init is needed really only the first time
ovs-vsctl --no-wait init

echo "Now start ovs-vswitchd (--no-ovsdb-server, since it is started already)"
ovs-ctl --no-ovsdb-server --db-sock="$DB_SOCK" start
sleep 5

#Max-Revalidator
#Specifies the maximum time (in ms) that revalidator threads
#will wait for kernel statistics before executing flow revalidation.
# Tweaking this value  is  discouraged
# unless you know exactly what you’re doing.

# ovs-vsctl set Open_vSwitch . other_config:max-revalidator=10000

#n-handler-threads
#Specifies the number of threads for software datapaths to use for handling new flows.
#The default value is the number of online CPU cores minus the number of revalidators.
ovs-vsctl set Open_vSwitch . other_config:n-handler-threads=4

#n-revalidator-threads
#Specifies the number of threads for software datapaths to use for revalidating flows in the datapath.
ovs-vsctl set Open_vSwitch . other_config:n-revalidator-threads=4

set +x

if [ $OVS_LOG_ENABLE == 'True' ]; then
echo "!!! WARNING !!!: Enabling OVS logging, please disable for performance testing"
enable_ovs_tc_logging
fi

ovs-ctl status
ovs-vsctl get Open_vSwitch . other_config

echo "================================================================================"
echo "================================================================================"
echo "Launch GDB to attach to ovs-vswitchd in another window: "
echo 'gdb $(which ovs-vswitchd) $(pidof ovs-vswitchd)'
read -r -p "Waiting 60 seconds, press any key to continue immediately" -t 60 -n 1 -s
echo "================================================================================"
echo "================================================================================"

}
```

Creating the OVS Bridges

To create the OVS bridges, use the following commands:

```
delete_ovs_bridges()
{
for bridge in $(ovs-vsctl list-br); do
ovs-vsctl del-br $bridge
done
}

setup_ovs_tc_bridges()
{
set +x
delete_ovs_bridges

set -x
ovs-vsctl add-br $OVS_BR0 -- set bridge $OVS_BR0 \
protocols=OpenFlow10,OpenFlow11,OpenFlow12,OpenFlow13,OpenFlow14
ifconfig $OVS_BR0 up
{ set +x; } 2>/dev/null
}
```

Adding PFs and Representor Ports to br0 and br1

To add PFs and representor ports to br0 and br1, use the following commands:

```
ovs_tc_add_vfreps_to_br0()
{
for vfr_if in `ls /sys/class/net`
do
if [ -d "/sys/class/net/$vfr_if/device" ]; then
# echo "HW device $vfr_if, skipping"
continue
fi
vfr_port_name=`cat /sys/class/net/$vfr_if/phys_port_name 2>/dev/null`
if [[ ! -z $vfr_port_name ]] && [[ $vfr_port_name == *"pf0"* ]]; then
echo "$vfr_if is a VF-rep of $vfr_port_name"
sudo ip link set dev $vfr_if down
sudo ip link set dev $vfr_if name $vfr_port_name
sudo ip link set dev $vfr_port_name up
sleep 1

ovs-vsctl --if-exists del-port $OVS_BR0 $vfr_port_name
set -x
ovs-vsctl add-port $OVS_BR0 $vfr_port_name
{ set +x; } 2>/dev/null
sleep 1
fi
done
}

setup_representor_tc_ports_br0()
{
export PF0=`ls -l /sys/class/net | grep ${PCI_PF0} | awk '{print $9}'`
if [ -z "$PF0"  ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: Could not find PF0."
```

```
echo "       To fix, please try to rebuild bnxt_en."
return
fi

if [ $SETUP_VXLAN == 'True' ]; then
local host=`echo $HOST_IP|cut -d "." -f 4`
ifconfig $PF0 172.20.1.${host}/24

ovs-vsctl --if-exists del-port $OVS_BR0 vxlan0
set -x
ovs-vsctl add-port $OVS_BR0 vxlan0 -- set Interface vxlan0 type=vxlan options:remote_ip=172.20.1.${VXLAN_VTEP}
 mtu_request=${OVS_MTU} options:key=${VXLAN_KEY0} options:dst_port=4789
{ set +x; } 2>/dev/null
else
ovs-vsctl --if-exists del-port $OVS_BR0 $PF0
set -x
ovs-vsctl add-port $OVS_BR0 $PF0
{ set +x; } 2>/dev/null
fi

ovs_tc_add_vfreps_to_br0
}
```

Configuring OVS-DPDK

The following scripts can be retrieved from the git repository referenced in Test Guidance. These functions are for reference only and may not be the latest versions. Copy and paste from the git repo if you need to use them.

Note: The following sequence of steps in BOLD are important. Specially binding trusted VFs to VFIO/IGB_UIO after the OVS bridges are created.

Setting up OVS

To setup OVS, use the following commands:

```
setup_ovs()
{
set -x
echo "Stopping OvS"
ovs-ctl stop
sleep 5
echo "Make sure OvS has stopped..."
ovs-ctl status

echo "Setup trusted VFs"
setup_trusted_vfs

if [ $USE_VFIO == 'True' ]; then
echo "Insert VFIO module"
load_vfio_module
else
echo "Insert IGB UIO module"
load_igb_uio_module
fi
```

```
echo "Setup hugepage mappings for NUMA systems"
ALLOCATE_HUGEPAGES='True'
set_numa_pages
ALLOCATE_HUGEPAGES='False'

echo "Start and initialize OvS"
ovs_db_init

echo "Create $OVS_BR0 and $OVS_BR1 bridges"
setup_ovs_bridges

#echo "Display current Ethernet/Baseband/Crypto device settings"
#show_devices

if [ $USE_VFIO == 'True' ]; then
echo "Bind Ethernet/Baseband/Crypto device to VFIO module"
bind_devices_to_vfio
else
echo "Bind Ethernet/Baseband/Crypto device to IGB UIO module"
bind_devices_to_igb_uio
fi

echo "Add trusted VFs and representor ports to $OVS_BR0 and $OVS_BR1"
setup_representor_ports_br0_br1
{ set +x; } 2>/dev/null

}
```

Creating VFs and Enabling Trust

To create VFs and enable trust, use the following commands:

```
setup_trusted_vfs()
{
set +x

# We want interfaces that own the IP to reply to ARP requests
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

load_bnxt_en_module
sleep 5
export PF0=`ls -l /sys/class/net | grep ${PCI_PF0} | awk '{print $9}'`
export PF1=`ls -l /sys/class/net | grep ${PCI_PF1} | awk '{print $9}'`

if [ -z "$PF0"    ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: Could not find PF0 and PF1."
echo "          To fix, please try to rebuild bnxt_en."
quit
return
fi

echo ""
```

```
echo "Bringing $PF0 and $PF1 links UP"
ip link set dev ${PF0} up
ip link set dev ${PF1} up
if [ $SETUP_VXLAN == 'False' ] || [ $SETUP_VXLAN_IPV6 == 'False' ]; then
local host=`echo $HOST_IP|cut -d "." -f 4`
ifconfig $PF0 172.20.1.${host}/24
ifconfig $PF0 inet6 add 2001::${host}/64 up
echo "PF0: $PF0 IP v4:172.20.1.${host} v6:2001::${host}"
ifconfig $PF1 172.30.1.${host}/24
ifconfig $PF1 inet6 add 3001::${host}/64 up
echo "PF1: $PF1 IP v4:172.30.1.${host} v6:3001::${host}"
echo ""
fi

sleep 1

if [ $CHECK_SRIOV -eq 0 ]; then
echo "!!! WARNING !!!: Bailing out VF creation..."
echo "                 To fix, please set CHECK_SRIOV to 1 in the override.conf."
return
fi

if [ -f /sys/class/net/${PF0}/device/sriov_totalvfs ]; then
tot_vfs=`cat /sys/class/net/${PF0}/device/sriov_totalvfs`
else
tot_vfs=0
fi
vfs=${NUMVFS_PF0:-$tot_vfs}
if [ $vfs -ne 0 -a $vfs -le $tot_vfs ]; then
#echo 0 > /sys/class/net/${PF0}/device/sriov_numvfs
echo $vfs > /sys/class/net/${PF0}/device/sriov_numvfs
sleep 2

echo "Setup IP and MTU on PF0 VFs"
setup_ip_on_pf0_vfs

if [ $OVS_TC_SUPPORT != 'true' ] && [ $LINUX_TC_SUPPORT != 'true' ]; then
  for ((vf=0; vf<${NUM_TRUSTED_VFS:-1}; vf++)); do
      echo "Setup TRUST ON PF0 VF $vf"
      ip link set dev ${PF0} vf ${vf} state enable
      ip link set dev ${PF0} vf ${vf} trust on
  done
fi

else
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## WARNING: Number of VFs on PF0"
echo "            To fix, please check NUMVFS_PF0 in override.conf."
fi

sleep 2
if [ -f /sys/class/net/${PF1}/device/sriov_totalvfs ]; then
tot_vfs=`cat /sys/class/net/${PF1}/device/sriov_totalvfs`
```

```
else
tot_vfs=0
fi
vfs=${NUMVFS_PF1:-$tot_vfs}
if [ $vfs -ne 0 -a $vfs -le $tot_vfs ]; then
#echo 0 > /sys/class/net/${PF1}/device/sriov_numvfs
echo $vfs > /sys/class/net/${PF1}/device/sriov_numvfs
sleep 2
echo "Setup IP and MTU on PF1 VFs"
setup_ip_on_pf1_vfs

if [ $OVS_TC_SUPPORT != 'true' ] && [ $LINUX_TC_SUPPORT != 'true' ]; then
  for ((vf=0; vf<${NUM_TRUSTED_VFS:-1}; vf++)); do
      echo "Setup TRUST ON PF1 VF ${vf}"
      ip link set dev ${PF1} vf ${vf} state enable
      ip link set dev ${PF1} vf ${vf} trust on
  done
fi
else
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## WARNING: Number of VFs on PF1"
echo "            To fix, please check NUMVFS_PF1 in override.conf."
fi

echo "VF interfaces ready for ping test "
#ip link show dev $PF0
#ip link show dev $PF1
{ set +x; } 2>/dev/null
}
```

Unloading VFIO Modules

To unload the VFIO modules, use the following commands:

```
remove_vfio_module()
{
echo "Unloading any existing VFIO module"
/sbin/lsmod | grep -s vfio > /dev/null
if [ $? -eq 0 ] ; then
sudo /sbin/rmmod vfio-pci
sudo /sbin/rmmod vfio_virqfd
sudo /sbin/rmmod vfio_iommu_type1
sudo /sbin/rmmod vfio
fi
}
```

Loading a New vfio-pci (and vfio Module if Required)

To load a new vfio-pci, and vfio module if required, use the following commands:

```
load_vfio_module()
{
remove_vfio_module

VFIO_PATH="kernel/drivers/vfio/pci/vfio-pci.ko*"
```

```
echo "Loading VFIO module"
/sbin/lsmod | grep -s vfio_pci > /dev/null
if [ $? -ne 0 ] ; then
if [ -f /lib/modules/$(uname -r)/$VFIO_PATH ] ; then
   sudo /sbin/modprobe vfio-pci
fi
fi

# make sure regular users can read /dev/vfio
echo "chmod /dev/vfio"
sudo chmod a+x /dev/vfio
if [ $? -ne 0 ] ; then
echo "FAIL"
quit
fi
echo "OK"

# check if /dev/vfio/vfio exists - that way we
# know we either loaded the module, or it was
# compiled into the kernel
if [ ! -e /dev/vfio/vfio ] ; then
echo "## ERROR: VFIO not found!"
fi
}
```

Starting OVS with DPDK

To start OVS with DPDK, use the following commands:

```
ovs_db_init()
{
echo "Remove the old ovs-vswitchd.log file"
cat /dev/null > /usr/local/var/log/openvswitch/ovs-vswitchd.log

if [ ! -f $OVS_DIR/utilities/ovs-ctl ];then
echo -n "$(basename $BASH_SOURCE)@$LINENO:"
echo "## ERROR: ovs-ctl not built."
echo "       To fix,  try to rebuild OvS."
return
fi

timeout 5 ovs-vsctl --if-exists del-br ${OVS_BR0}-vhost-forwarder
timeout 5 ovs-vsctl --if-exists del-br ${OVS_BR1}-vhost-forwarder
timeout 5 ovs-vsctl --if-exists del-br ${OVS_BR0}-vxlan
timeout 5 ovs-vsctl --if-exists del-br ${OVS_BR1}-vxlan
timeout 5 ovs-vsctl --if-exists del-br $OVS_BR0
timeout 5 ovs-vsctl --if-exists del-br $OVS_BR1
timeout 5 ovs-ctl stop
sleep 5
echo "Make sure OvS has stopped..."
ovs-ctl status
sleep 1
```

```
echo "First start ovsdb server without ovs-vswitchd"
ovs-ctl --no-ovs-vswitchd start

#per-port-memory=true needs to happen before dpdk-init=true
if [ $NUMVFS -le 16 ]; then
ovs-vsctl set Open_vSwitch . other_config:per-port-memory=true
else
# For scaling it creates mempool per port and not shared
ovs-vsctl set Open_vSwitch . other_config:per-port-memory=false
fi
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-mask=$(comma_list_to_hex)
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=$(comma_list_to_hex `echo $OVS_L-
CORE_MASK`)
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-mem-channels=4
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem=$OVS_SOCKET_MEM
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-extra="$OVS_DPDK_EXTRA"
ovs-vsctl set Open_vSwitch . other_config:hw-offload=$OVS_HW_OFFLOAD
sleep 1

#ovs-vsctl --no-wait set Open_vSwitch . other_config:emc-insert-inv-prob=0
# This init is needed really only the first time
ovs-vsctl --no-wait init

echo "Now start ovs-vswitchd (--no-ovsdb-server, since it is started already)"
ovs-ctl --no-ovsdb-server --db-sock="$DB_SOCK" start
sleep 5

if [ $OVS_LOG_ENABLE == 'True' ]; then
echo "!!! WARNING !!!: Enabling OVS logging,  disable for performance testing"
enable_ovs_logging
fi

ovs-ctl status
echo -n "Open_vSwitch other_config:dpdk_initialized is: "
ovs-vsctl get Open_vSwitch . dpdk_initialized
# Additionally, the library version linked to ovs-vswitchd can be confirmed
# with either the ovs-vswitchd logs, or by running either of the commands:
# ovs-vswitchd --version
echo -n "Open_vSwitch other_config:dpdk_version is: "
ovs-vsctl get Open_vSwitch . dpdk_version

echo "================================================================================"
echo "================================================================================"
echo "Launch GDB to attach to ovs-vswitchd in another window: "
echo 'gdb $(which ovs-vswitchd) $(pidof ovs-vswitchd)'
read -r -p "Waiting 60 seconds, press any key to continue immediately" -t 60 -n 1 -s
echo "================================================================================"
echo "================================================================================"

}
```

Additional CPU Mask Argument Information

This section provides additional information on the CPU Mask Arguments shown in the previous sections.

See https://developers.redhat.com/blog/2017/06/28/ovs-dpdk-parameters-dealing-with-multi-numa#dpdk_lcore_mask for additional information.

1. To create br0 and br1 bridges, use the following commands:

```
ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev protocols=OpenFlow10,OpenFlow11,Open-
Flow12,OpenFlow13,OpenFlow14
ovs-vsctl add-br br1 -- set bridge br1 datapath_type=netdev protocols=OpenFlow10,OpenFlow11,Open-
Flow12,OpenFlow13,OpenFlow14
```

2. To bind the Ethernet/Baseband/Crypto device to VFIO module, using the following commands:

```
PCI_PATH=B:D.F
sudo python3 ${RTE_SDK}/usertools/dpdk-devbind.py --force -b vfio-pci $PCI_PATH
```

3. To add trusted VFs and representor ports to br0 using the following commands:

```
ovs-vsctl add-port br0 pf0_vf0 -- set Interface pf0_vf0 type=dpdk options:dpdk-devargs=0000:b4:00.0
 mtu_request=1900 options:n_rxq=2
```

Note: This is optional for performance tuning and so forth. OvS should balance the rings.

```
ovs-vsctl set interface pf0_vf0 options:n_rxq=2 other_config:pmd-rxq-affinity=0:1,1:2
ovs-vsctl add-port br0-vxlan vxlan0 -- set Interface vxlan0 type=vxlan options:remote_ip=172.20.1.46
 mtu_request=1900 options:key=flow options:dst_port=4789
```

Note: Syntax “representor= <vfid0 based for the PF> VF 0 we usually use it for trusted to VF 1 representor will have representor=1, VF 2 will be representor=2 and so on.

```
ovs-vsctl add-port br0-vxlan enp180s0f1_1 -- set Interface enp180s0f1_1 type=dpdk options:dpdk-de-
vargs=0000:b4:00.0,representor=1 mtu_request=1900 options:n_rxq=2
ovs-vsctl set interface enp180s0f1_1 options:n_rxq=2 other_config:pmd-rxq-affinity=0:1,1:2
ovs-vsctl add-port br0-vxlan enp180s0f2_2 -- set Interface enp180s0f2_2 type=dpdk options:dpdk-de-
vargs=0000:b4:00.0,representor=2 mtu_request=1900 options:n_rxq=2
ovs-vsctl set interface enp180s0f2_2 options:n_rxq=2 other_config:pmd-rxq-affinity=0:1,1:2
```

4. To add trusted VFs and representor ports to br1, use the following commands:

```
ovs-vsctl add-port br1 pf1_vf0 -- set Interface pf1_vf0 type=dpdk options:dpdk-devargs=0000:b4:01.0
 mtu_request=1900 options:n_rxq=2
ovs-vsctl set interface pf1_vf0 options:n_rxq=2 other_config:pmd-rxq-affinity=0:3,1:4
++ ovs-vsctl add-port br1-vxlan vxlan1 -- set Interface vxlan1 type=vxlan options:remote_ip=172.30.1.46
 mtu_request=1900 options:key=flow options:dst_port=4789
ovs-vsctl add-port br1-vxlan enp180s1f1_1 -- set Interface enp180s1f1_1 type=dpdk options:dpdk-de-
vargs=0000:b4:01.0,representor=1 mtu_request=1900 options:n_rxq=2
ovs-vsctl set interface enp180s1f1_1 options:n_rxq=2 other_config:pmd-rxq-affinity=0:3,1:4
ovs-vsctl add-port br1-vxlan enp180s1f2_2 -- set Interface enp180s1f2_2 type=dpdk options:dpdk-de-
vargs=0000:b4:01.0,representor=2 mtu_request=1900 options:n_rxq=2
ovs-vsctl set interface enp180s1f2_2 options:n_rxq=2 other_config:pmd-rxq-affinity=0:3,1:4
```

OVS Flow Offloads

Provides and overview of various OVS modes and configuration information for full offload (VF representors)use cases.

OVS Operation Modes and Sample Flow Add Commands

- **OVS Normal Mode (fail-mode=standalone)**
  This is the default mode in which OVS acts like a regular Layer-2 learning switch. OVS starts out with a single rule in its OpenFlow tables:

```
ovs-ofctl dump-flows ovsbr0
```

Output:

```
cookie=0x0, duration=6.144s, table=0, n_packets=65, n_bytes=6314, actions=NORMAL
```

The normal action indicates that packets should be handled like a normal learning switch. OVS adds datapath rules based on this learning, when packets start flowing through the switch. The normal mode is configured for a bridge by default if the “fail-mode” option is unspecified or if it is set to “standalone” while creating the bridge using ovs-vsctl command. For example:

```
ovs-vsctl add-br ovsbr0  -- set bridge ovsbr0
```

The OpenFlow Classifier implementation in OVS generates datapath flows that are megaflows. Megaflows are flows with wildcarded fields. Currently the Broadcom TruFlow implementation does not support offloading flows with partial WC masks for IPv6. To minimize conditions that could generate megaflows with WC bits, it is suggested that the normal mode be disabled while using explicit flow creation using ovs-ofctl. That is, avoid simultaneously using the normal mode and explicit flow configuration using ovs-ofctl. If explicit flow configuration is needed, disable normal mode using fail-mode=secure, as described in the next section.

- **OVS Normal Mode Disabled (fail-mode=secure)**

  In this mode, OVS requires OpenFlow rules to be configured to be able to forward packets. The OpenFlow rules can be provided through a separate OpenFlow Controller or can be configured through the ovs-ofctl CLI command in the server. In this mode, OVS starts out with its OpenFlow tables empty. When a packet enters a port in OVS, if an OpenFlow rule exists to process the packet, then, OVS adds a corresponding datapath rule and forwards the packet. Otherwise, the default action is to drop the packet and the packet will not be able to get through OVS.The normal mode can be disabled and the secure mode can be enabled, if the “fail-mode” option is set to “secure” while creating the bridge using ovs-vsctl command.

  ```
  ovs-vsctl add-br ovsbr0  -- set bridge ovsbr0 fail-mode=secure
  ```

- **OVS Flow Add example for 5-tuple Match with VxLAN and Inner IPv4**

  ```
  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${ovs_port0} dl_type=0x0800 ip_proto=6 tcp,
   nw_src=192.160.1.${VXLAN_VTEP} nw_dst=192.160.1.${host} tp_src=${i} tp_dst=${i} actions=output:${VF}_
  ${vfid}"

  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${VF}_${vfid} dl_type=0x0800 ip_proto=6
   tcp, nw_src=192.160.1.${host} nw_dst=192.160.1.${VXLAN_VTEP} tp_src=${i} tp_dst=${i} actions=out-
  put:${ovs_port0}"
  ```

- **OVS Flow Add example for 5-tuple Match with VxLAN and Inner IPv6**

  ```
  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${ovs_port0} dl_type=0x86dd ip_proto=6,
   ipv6_src=::192.160.1.${VXLAN_VTEP} ipv6_dst=::192.160.1.${host} tp_src=${i} tp_dst=${i} actions=out-
  put:${VF}_${vfid}"

  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${VF}_${vfid} dl_type=0x86dd ip_proto=6,
   ipv6_src=::192.160.1.${host} ipv6_dst=::192.160.1.${VXLAN_VTEP} tp_src=${i} tp_dst=${i} actions=out-
  put:${ovs_port0}"
  ```

- **OVS Flow Add Example Inner IPv4 and IPv6 to Insert Tunnel ID**

  ```
  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${VF}_${vfid} dl_type=0x0800 ip_pro-
  to=${ovs_ip_proto}, nw_src=192.160.1.${host} nw_dst=192.160.1.${VXLAN_VTEP} tp_src=${i} tp_dst=${i} action-
  s=set_field:${i}->tun_id,resubmit(,1)"

  ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${VF}_${vfid} dl_type=0x86dd ip_pro-
  to=${ovs_ip_proto}, ipv6_src=::192.160.1.${host} ipv6_dst=::192.160.1.${VXLAN_VTEP} tp_src=${i} tp_dst=${i}
   actions=set_field:${i}->tun_id,resubmit(,1)"

  ovs-ofctl add-flow $ovs_br0 "table=1, priority=65535 actions=output:${ovs_port0}"
  ```

- **OVS Flow Add Example Inner IPv4 with NAT Action**

```
local snat="192.170.1.${host}"
local spat="$((i+num_flows))"
local dnat="192.170.1.${VXLAN_VTEP}"
local dpat="$((i+num_flows))"

unset ing_nat_action
unset egr_nat_action
ing_nat_action="mod_nw_dst:${dnat},mod_tp_dst:${dpat},"
egr_nat_action="mod_nw_dst:${snat},mod_tp_dst:${dpat},"

ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${ovs_port0} dl_type=0x0800 ip_pro-
to=${ovs_ip_proto}, nw_src=192.160.1.${VXLAN_VTEP} nw_dst=192.160.1.${host} tp_src=${i} tp_dst=${i} ac-
tions=${ing_nat_action}output:${VF}_${vfid}"

ovs-ofctl add-flow $ovs_br0 "table=0, priority=65535 in_port=${VF}_${vfid} dl_type=0x0800 ip_pro-
to=${ovs_ip_proto}, nw_src=${dnat} nw_dst=192.160.1.${VXLAN_VTEP} tp_src=${dpat} tp_dst=${i} ac-
tions=${egr_nat_action}output:${ovs_port0}"
```

- **TC Flow Add Example VxLAN Decap/Encap**

```
tc filter add dev <PF0> protocol ip parent ffff: flower skip_sw src_mac 00:01:22:33:81:60 dst_mac
 b0:26:28:e0:4b:30 enc_src_ip 20.20.21.2 enc_dst_ip 20.20.21.1 enc_dst_port 4789 enc_key_id 22 action tun-
nel_key unset action mirred egress redirect dev <VFR0>

tc filter add dev <VFR0> protocol ip parent ffff: flower skip_sw src_mac 00:11:22:33:55:66 dst_mac
 00:10:19:ad:1b:11 action tunnel_key set src_ip 20.20.21.1 dst_ip 20.20.21.2 dst_port 4789 id 22 action
 mirred egress redirect dev vxlan0

#Add TC rules using the following format:
tc filter add dev NETDEVICE ingress protocol PROTOCOL prio PRIORITY [chain CHAIN] flower [MATCH_LIST] [ac-
tion ACTION_SPEC]

#Display TC rules using the following format:
tc [-s] filter show dev NETDEVICE ingress
```

Note: See Test Guidance to obtain the actual scripts to see the details, above only gives examples to give a feel for what to do.

Test Guidance

Provides test guidance for OVS and TC/DPDK offload information.

OVS Reference

See https://docs.openvswitch.org/en/latest/contents/ for the latest build, setup, and usage instructions.

OVS Aliases

The following shell script and source can be used for debugging OvS.

- `ovs-alias.sh`

```
#!/bin/bash
if [ -z "$PS1" ] ; then
```

```
echo -e "This script must be sourced. Use \"source <script>\" instead."
exit
fi

#echo " Sourcing OvS Aliases...."
alias cpufreq='watch -n.1 "cat /proc/cpuinfo | grep \"^[c]pu MHz\""'

alias python='python3'
alias grep='grep --color'

unset BREAK
export BREAK='if [ -z $LOOP ]; then break; fi'

export PATH=$PATH:$OVS_DIR/utilities:/usr/share/openvswitch/scripts
if [ -d /usr/local ]; then
USR_LOCAL="/usr/local"
fi
# Maia ssh
alias sshmaia='ssh -X root@192.168.1.10 -t "sudo bash" '
alias db-log='ovsdb-tool -mm show-log ${USR_LOCAL}/etc/openvswitch/conf.db'
alias db-cldi='ovsdb-client dump Interface'

# Port stats simple view, takes bridge as input
alias of-dp='ovs-ofctl -O OpenFlow14 dump-ports '
alias of-wdp="watch -n 1 -d 'ovs-ofctl -O OpenFlow14 dump-ports'"

# OpenFlow port/flow dump
alias of-sh='ovs-ofctl show '
alias of-df='ovs-ofctl dump-flows '

# hidden flows
alias app-brdf='ovs-appctl bridge/dump-flows '

# DP flows, shows only active flows
alias app-dpdf='ovs-appctl dpif/dump-flows '
alias app-wdpdfbr0="watch -n 1 -d 'ovs-appctl dpif/dump-flows br0'"
alias app-wdpdfbr1="watch -n 1 -d 'ovs-appctl dpif/dump-flows br1'"

alias app-dpsh='ovs-appctl dpif/show'
alias app-wperf="watch -n 1 -d 'ovs-appctl dpif-netdev/pmd-perf-show'"
alias app-perf='ovs-appctl dpif-netdev/pmd-perf-show'
alias app-rxq='ovs-appctl dpif-netdev/pmd-rxq-show'
alias app-cov='ovs-appctl coverage/show'
alias app-stats='ovs-appctl dpif-netdev/pmd-stats-show'
alias app-stats10='ovs-appctl dpif-netdev/pmd-stats-clear  sleep 10s && ovs-appctl dpif-netdev/pmd-stats-
show'
alias app-cstats='ovs-appctl dpif-netdev/pmd-stats-clear'
alias app-wstats="watch -n 1 -d 'ovs-appctl dpif-netdev/pmd-stats-show'"
alias app-cycles='ovs-appctl dpif-netdev/pmd-stats-show|egrep pmd\|cycle'
alias app-wcycles="watch -n 1 -d 'ovs-appctl dpif-netdev/pmd-stats-show|egrep pmd\|cycle'"
```

- Enabling Custom Counters

```
alias app-wcov="watch -d -n1 'ovs-appctl coverage/show | egrep \"forward|netdev_rece|netdev_sent\"'"
alias app-cov='ovs-appctl coverage/show'
```

- Data Path Flows

```
alias app-dpdfm='ovs-appctl dpctl/dump-flows -m'
alias app-wdpdfm="watch -n 1 -d 'ovs-appctl dpctl/dump-flows -m'"
alias tf-vxlan-offd="while [ "1" ];\
do \
echo 'F1 Offloaded                          :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_pop(vxlan_sys_4789)';\
echo  "------------------------------------------------------------";\
echo 'F2-ipv4 Offloaded                     :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x0800)';\
echo  "------------------------------------------------------------";\
echo 'F2-ipv6 Offloaded                     :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x86dd)';\
echo  "------------------------------------------------------------";\
echo 'Encap-ipv4 Offloaded                  :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x0800)';\
echo  "------------------------------------------------------------";\
echo 'Encap-ipv6 Offloaded                  :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x86dd)';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-vxlan-offc="while [ "1" ];\
do \
echo -n 'F1 Ofloaded                        :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_pop(vxlan_sys_4789)'|wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'F2-ipv4 Offloaded                   :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x0800)' |
wc -l;\
echo -n 'F2-ipv6 Offloaded                   :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x86dd)' |
wc -l;\
echo -n 'F2-Total Offloaded                  :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'Encap-ipv4 Offloaded                :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x0800)' |
wc -l;\
echo -n 'Encap-ipv6 Offloaded                :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x86dd)' |
wc -l;\
echo -n 'Encap-Total Offloaded               :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'Total Offloaded                    :' ;\
```

```
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |wc -l;\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-vxlan-missd="while [ "1" ];\
do \
echo 'Miss F1                              :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_pop(vxlan_sys_4789)';\
echo  "------------------------------------------------------------";\
echo 'Miss F2-ipv4                         :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep
 'eth_type(0x0800)';\
echo  "------------------------------------------------------------";\
echo 'Miss F2-ipv6                         :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x86d-
d)';\
echo  "------------------------------------------------------------";\
echo 'Miss Encap-ipv4                      :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep
 'eth_type(0x0800)';\
echo  "------------------------------------------------------------";\
echo 'Miss Encap-ipv6                      :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x86d-
d)';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-vxlan-missc="while [ "1" ];\
do \
echo -n 'Miss F1                           :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_pop(vxlan_sys_4789)'|wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'Miss F2-ipv4                      :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x0800)'
 |wc -l;\
echo -n 'Miss F2-ipv6                      :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'in_port(vxlan_sys_4789)' |grep 'eth_type(0x86dd)'
 |wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'Miss Encap-ipv4                   :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep
 'eth_type(0x0800)' |wc -l;\
echo -n 'Miss Encap-ipv6                   :' ;\
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep 'tnl_port(vxlan_sys_4789)' |grep 'eth_type(0x86d-
d)' |wc -l;\
echo  "------------------------------------------------------------";\
echo -n 'Miss Total                        :' ;\
```

```
app-dpdfm |grep -v 'icmp' |grep -v 'offloaded:yes' |grep -v 'tnl_pop(vxlan_sys_4789)' |grep -v 'flow-dump'
 |wc -l;\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-vxlan="\
tf-vxlan-offd;\
tf-vxlan-missd;\
tf-vxlan-offc;\
tf-vxlan-missc;\
"
alias tf-vxland="\
tf-vxlan-offd;\
tf-vxlan-missd;\
"
alias tf-vxlanc="\
tf-vxlan-offc;\
tf-vxlan-missc;\
"
alias tf-offc="while [ "1" ];\
do app-dpdfm |grep 'offloaded:yes' |wc -l;\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK;\
sleep 1;\
done"
alias tf-missc="while [ "1" ];\
do app-dpdfm |grep -v 'offloaded:yes' |wc -l;\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-missd="while [ "1" ];\
do app-dpdfm |grep -v 'offloaded:yes';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-offd="while [ "1" ];\
do app-dpdfm |grep -E 'offloaded:yes';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
```

```
done"
alias tf-arp="while [ "1" ];\
do app-dpdfm |grep -E '0x0806';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-ip="while [ "1" ];\
do \
echo -n 'ipv4                 :' ;\
app-dpdfm |grep 'eth_type(0x0800)';\
echo -n 'ipv6                 :' ;\
app-dpdfm |grep 'eth_type(0x86dd)';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-ip-offd="while [ "1" ];\
do \
echo -n 'ipv4                 :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep -v 'tnl_pop(vxlan_sys_4789)' |grep
 'eth_type(0x0800)';\
echo -n 'ipv6                 :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep -v 'tnl_pop(vxlan_sys_4789)' |grep 'eth_type(0x86d-
d)';\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-ip-offc="while [ "1" ];\
do \
echo -n 'ipv4                 :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep -v 'tnl_pop(vxlan_sys_4789)' |grep 'eth_type(0x0800)'
 |wc -l;\
echo -n 'ipv6                 :' ;\
app-dpdfm |grep -v 'icmp' |grep 'offloaded:yes' |grep -v 'tnl_pop(vxlan_sys_4789)' |grep 'eth_type(0x86dd)'
 |wc -l;\
echo  "------------------------------------------------------------";\
date && date +%s;\
echo  "============================================================";\
$BREAK
sleep 1;\
done"
alias tf-icmp="while [ "1" ];\
do app-dpdfm |grep -E 'icmp\(';\
echo  "------------------------------------------------------------";\
date && date +%s;\
```

```
echo  "================================================================";\
$BREAK
sleep 1;\
done"

alias tf-icmpv6="while [ "1" ];\
do app-dpdfm |grep -E 'icmpv6\(';\
echo  "----------------------------------------------------------------";\
date && date +%s;\
echo  "================================================================";\
$BREAK
sleep 1;\
done"

alias tf-icmp-offd="while [ "1" ];\
do app-dpdfm |grep -E 'icmp\(' |grep 'offloaded:yes' ;\
echo  "----------------------------------------------------------------";\
date && date +%s;\
echo  "================================================================";\
$BREAK
sleep 1;\
done"

alias tf-icmpv6-offd="while [ "1" ];\
do app-dpdfm |grep -E 'icmpv6\(' |grep 'offloaded:yes' ;\
echo  "----------------------------------------------------------------";\
date && date +%s;\
echo  "================================================================";\
$BREAK
sleep 1;\
done"
```

- Logging>

```
alias app-llog='ovs-appctl vlog/list '
alias app-slog='ovs-appctl vlog/set dbg '
```

- List All appctl Commands

```
alias app-list='ovs-appctl list-commands'
#ARP
alias app-tas='ovs-appctl tnl/arp/show'
```

- Dump All Port Stats Including Bridges

```
alias app-dpsh='ovs-appctl dpctl/show -s '
alias app-wdpsh="watch -d -n1 'ovs-appctl dpctl/show -s|egrep -A4 -E dpdk\|dpdkvhostuser'"
```

- Mempool

```
alias app-mem='ovs-appctl netdev-dpdk/get-mempool-info '

#vSwitch information
alias vs-livs='ovs-vsctl list Open_vSwitch'
alias shvs='ovs-vsctl show'
alias vs-sh='ovs-vsctl show'
```

```
alias vs-liin='ovs-vsctl list interface '
alias vs-lipo='ovs-vsctl list port '
alias vs-libr='ovs-vsctl list bridge '
```

- Set max-idle time

```
function vs-sidle()
{
if [ -z $1 ]; then
 echo "usage: $0 <idle-time> "
 return 1
fi
ovs-vsctl --no-wait set Open_vSwitch . other_config:max-idle=$1
}
```

- vs-stats port

```
function vs-stats()
{
if [ -z $1 ]; then
 echo "usage: $0 <port> "
 return 1
fi
ovs-vsctl get interface $1 statistics
}
```

- vs-wstats port

```
function vs-wstats()
{
if [ -z $1 ]; then
   echo "usage: $0 <port1 port2 port3 ... >"
   return 1
fi
cmd=""
for port in "$@"; do
   cmd+="echo ========PORT:$port========  ovs-vsctl get interface $port statistics && "
done
cmd=`echo $cmd | rev | sed 's/&&/ /' | rev`
echo "CMD: $cmd"
watch -n1 -d "$cmd"
}
```

- vs-hstats port

```
function vs-hstats()
{
if [ -z $1 ]; then
    echo "usage: $0 <port> "
    return 1
fi
ovs-vsctl get interface $1 statistics|sed -e "s/,/\n/g" -e "s/[\",\{,\}, ]//g" -e "s/=/ =? /g"
}
function vs-whstats()
{
 if [ -z $1 ]; then
```

```
        echo "usage: $0 <port1 port2 port3 ...> "
        return 1
    fi
    cmd=""
    for port in "$@"; do
        cmd+="echo ========PORT:$port========  ovs-vsctl get interface $port statistics|sed -e 's/,/\n/g' -e
    's/[\",\{,\}, ]//g' -e 's/=/ =? /g' && "
    done
    cmd=`echo $cmd | rev | sed 's/&&/ /' | rev`
    echo "CMD: $cmd"
    watch -n1 -d "$cmd"
    #watch -n 1 -d "ovs-vsctl get interface $1 statistics|sed -e 's/,/\n/g' -e 's/[\",\{,\}, ]//g' -e 's/=/
    =? /g'"
}
```

- Logging

```
alias logt='tail -f ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
#ctrl+c, shift+f, q
alias logl='less +F ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logo='vi ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logc='cat ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logport='grep "ULP Port:" ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logflow='grep "rte_flow" ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logclear='cat /dev/null > ${USR_LOCAL}/var/log/openvswitch/ovs-vswitchd.log'
alias logtvhuser-port0='sudo tail -f /var/log/libvirt/qemu/vhuser-port0.log'
alias logtvhuser-port1='sudo tail -f /var/log/libvirt/qemu/vhuser-port1.log'
```

- gdb

```
alias ovsgdb='gdb $(which ovs-vswitchd) $(pidof ovs-vswitchd)'
```

- Monitor threads

```
alias ovstop='top -p `pidof ovs-vswitchd` -H -d1'
alias qemupidstat='pidstat -t -p `pidof qemu-system-x86_64` 1'
alias ovspidstat='pidstat -t -p `pidof ovs-vswitchd` 1'
```

- VM Related

```
function start-vm()
{
  systemctl is-active -q libvirtd.service && echo "libvirtd already started" || systemctl restart libvirt-
d.service
  virt-manager
  virsh list --all
  echo "HELP: virsh <list | start | shutdown> vm-name"
}
```

- Help

```
#alias ovshelp='alias | grep -E "ovs-|tf-" && declare -F | grep vs-'
alias ovs-help='cat $SCRIPTS/ovs-alias.sh'
```

Hardware and Regulatory Information

Provides information on Ethernet network adapter hardware and regulatory information.

This section contains the following information:

- Ethernet Network Adapter Functional Descriptions
- Ethernet Network Adapter Network Link and Activity LEDs
- Ethernet Network Adapter Regulatory and Safety Approvals

Ethernet Network Adapter Functional Descriptions

Provides the functional descriptions for each Ethernet network adapter.

Note: The following sections show the network interface cards in the BCM95719, BCM95720, BCM9574XX, BCM95750X, and BCM957608 families. The surface markings of the components and the bracket may not reflect the product upon receipt. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

BCM95719A1904AC Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM95719A1904AC network adapter and other similar 4x1G Base-T network adapters such as the BCM95719-P41TDF0S, BCM95719-P41TDL0S, BCM95719-P41TGF0S, and BCM95719-P41TGL0S.

Table 96: BCM95719A1904AC Functional Descriptions

Function	Description
Speed	Quad-Port 1 Gb/s Ethernet
PCIe	Gen2 x4. The NIC supports PCIe Gen2 and 1 speeds.
	The NIC supports PCIe Gen2 and 1 speeds. However, PCIe Gen2 is recommended to achieve nominal throughput when 4 ports of 1G links transmit and receive traffic at the same time.
Interface	RJ-45 for 1 Gb/s
Device	Broadcom BCM5719 Quad-Port 1GBASE-T PCIe 2.1 Ethernet Controller.
NDIS Name	Broadcom NetXtreme Gigabit Ethernet
UEFI Name	Broadcom NetXtreme Gigabit Ethernet (BCM5719)

Figure 43: BCM95719A1904AC Network Interface Card

BCM95719N1905C Ethernet Network Adapter

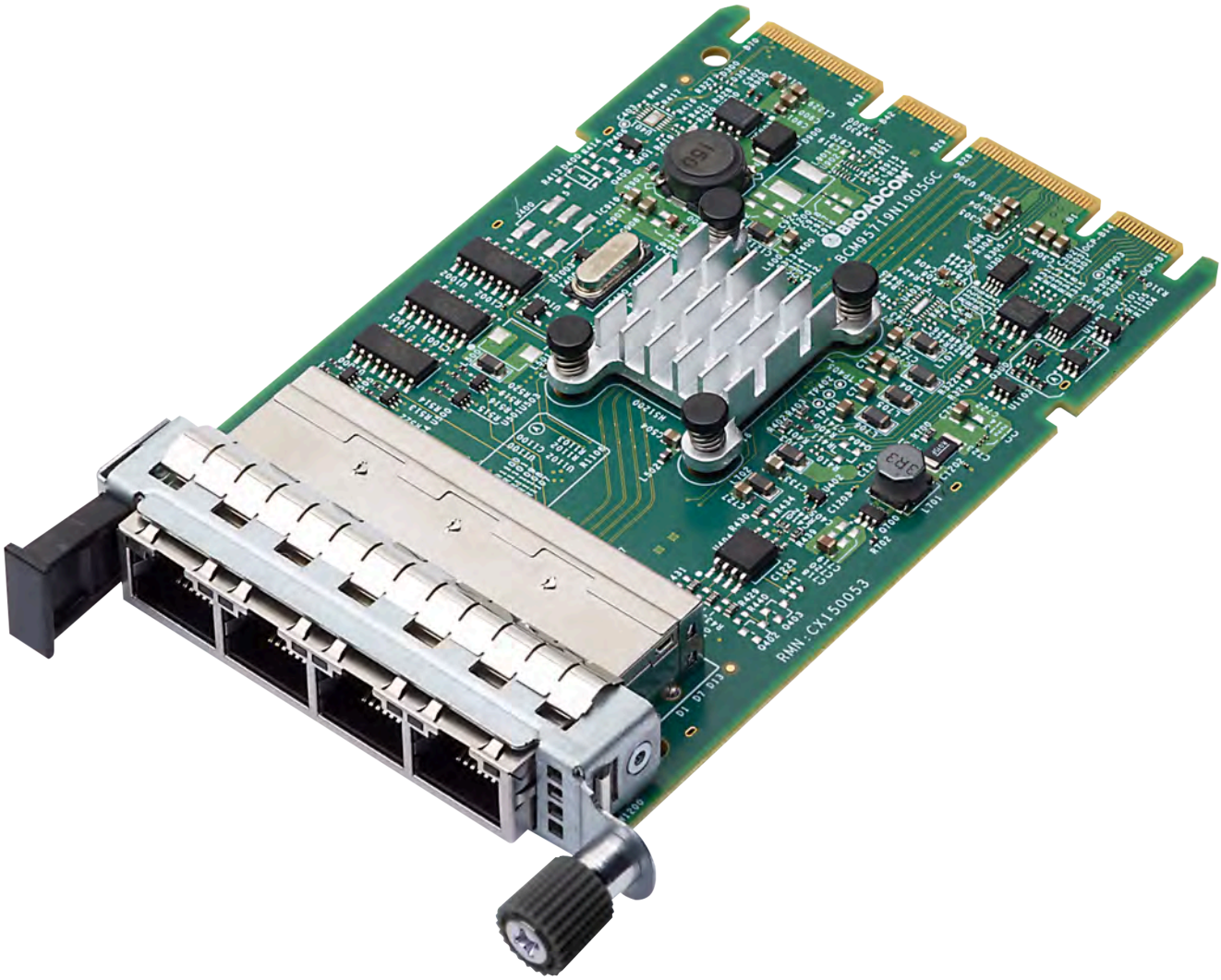
The information in the following table applies to the Broadcom BCM95719N1905C network adapter and other similar 4x1G Base-T network adapters such as the BCM95719-N41TDI0S and BCM95719-N41TGI0S.

Table 97: BCM95719N1905C Functional Descriptions

Function	Description
Speed	Quad-Port 1 Gb/s Ethernet
PCIe	Gen2 x4. The NIC supports PCIe Gen2 and 1 speeds.

Function	Description
	The NIC supports PCIe Gen2 and 1 speeds. However, PCIe Gen2 is recommended to achieve nominal throughput when 4 ports of 1G links transmit and receive traffic at the same time.
Interface	RJ-45 for 1 Gb/s
Device	Broadcom BCM5719 Quad-Port 1GBASE-T PCIe 2.1 Ethernet Controller.
NDIS Name	Broadcom NetXtreme Gigabit Ethernet
UEFI Name	Broadcom NetXtreme Gigabit Ethernet (BCM5719)

Figure 44: BCM95719N1905C Network Interface Card



BCM95720A2003AC Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM95720A2003AC network adapter.

Table 98: BCM95720A2003AC Functional Descriptions

Function	Description
Speed	Dual-Port 1Gb/s Ethernet
PCIe	Gen2 x1
	The NIC supports PCIe Gen2 and 1 speeds. However, PCIe Gen2 is recommended to achieve nominal throughput when 2 ports of 1G links transmit and receive traffic at the same time.
Interface	RJ-45 for 1 Gb/s
Device	Broadcom BCM5720 Dual-Port 1GBASE-T PCIe 2.1 Ethernet Controller.
NDIS Name	Broadcom NetXtreme Gigabit Ethernet
UEFI Name	Broadcom NetXtreme Gigabit Ethernet (BCM5720)

Figure 45: BCM95720A2003AC Network Interface Card

BCM957412A4120AC Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957412A4120AC network adapter.

Table 99: BCM957412A4120AC Functional Descriptions

Function	Description
Speed	Dual-Port 10 Gb/s Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 10G links transmit and receive traffic at the same time.

Function	Description
Interface	SFP+ for 10 Gb/s
Device	Broadcom BCM57412 10 Gb/s MAC controller with integrated dual-channel 10 Gb/s SFP+ transceiver.
NDIS Name	Broadcom P210p NetXtreme-E Dual-Port 10Gb Ethernet PCIe Adapter
UEFI Name	Broadcom P210p NetXtreme-E Dual-Port 10Gb Ethernet PCIe Adapter

Figure 46: BCM957412A4120AC Network Interface Card

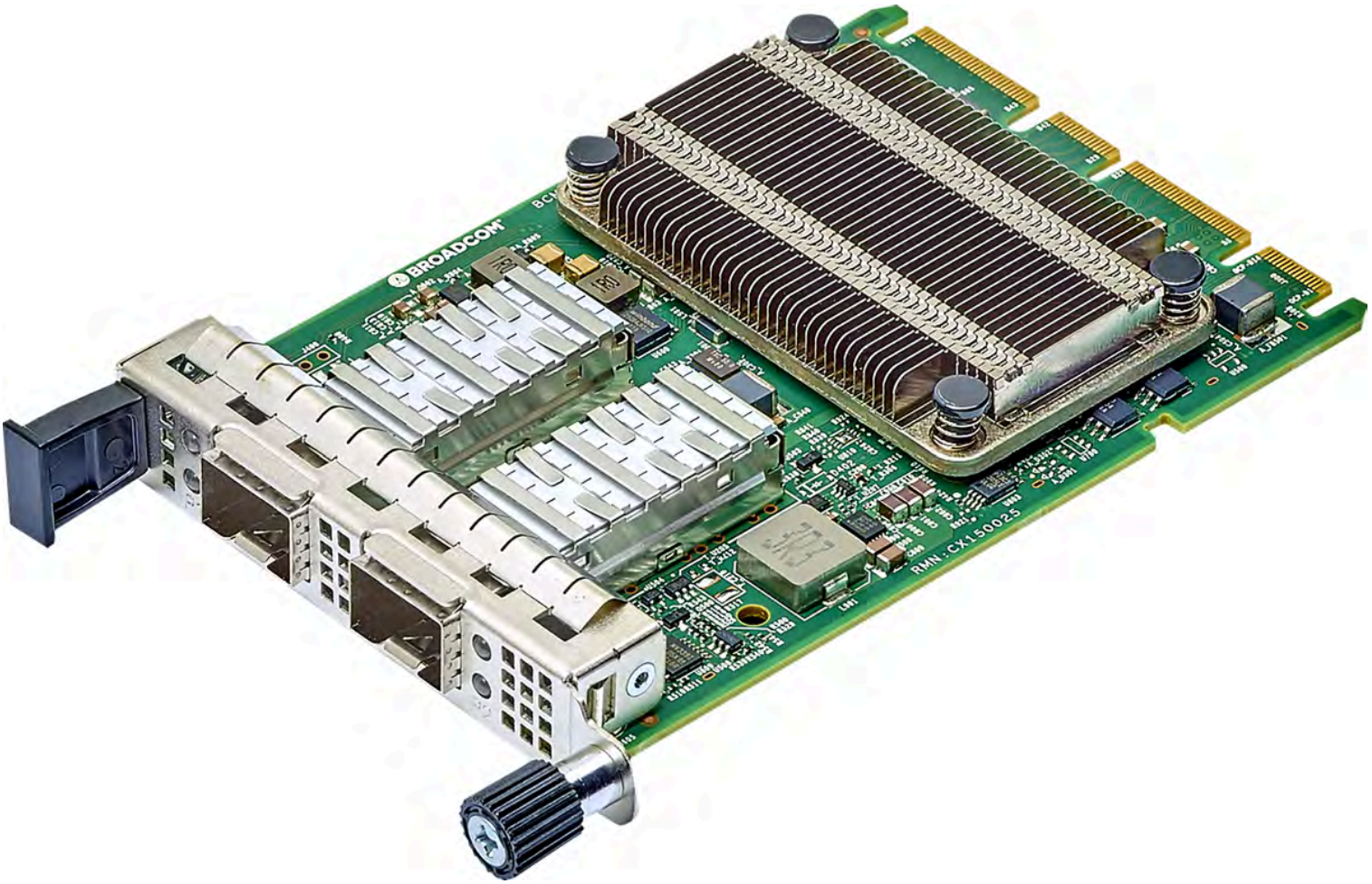
BCM957412N4120C Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957412N4120C network adapter.

Table 100: BCM957412N4120C Functional Descriptions

Function	Description
Speed	Dual-Port 10 Gb/s Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 25G links transmit and receive traffic at the same time.
Interface	SFP+ for 10 Gb/s
Device	Broadcom BCM57412 10 Gb/s MAC controller with integrated dual-channel 10 Gb/s SFP+ transceiver.
NDIS Name	Broadcom NetXtreme-E Series Dual-Port 10Gb SFP+ Ethernet OCP 3.0 Adapter
UEFI Name	Broadcom NetXtreme-E Dual 10Gb SFP+ OCP 3.0 Ethernet

Figure 47: BCM957412N4120 Small-Form-Factor Network Adapter



BCM957414A4142CC Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957414A4142CC network adapter and other similar 2x25G network adapters such as the BCM957414-P225DF0S, BCM957414-P225GF0S, BCM957414-P225DL0S, and BCM957414-P225GF0S.

Table 101: BCM957414A4142CC Functional Descriptions

Function	Description
Speed	Dual-Port 25 Gb/s Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 25G links transmit and receive traffic at the same time.
Interface	SFP28 for 25 Gb/s
Device	Broadcom BCM57414 25 Gb/s MAC controller with integrated dual-channel 25 Gb/s SFP28 transceiver.
NDIS Name	Broadcom P225p NetXtreme-E Dual-Port 10Gb/25Gb Ethernet PCIe Adapter
UEFI Name	Broadcom P225p NetXtreme-E Dual-Port 10Gb/25Gb Ethernet PCIe Adapter

Figure 48: /BCM957414A4142CC Network Interface Card

BCM957414N4140C Ethernet Network Adapter

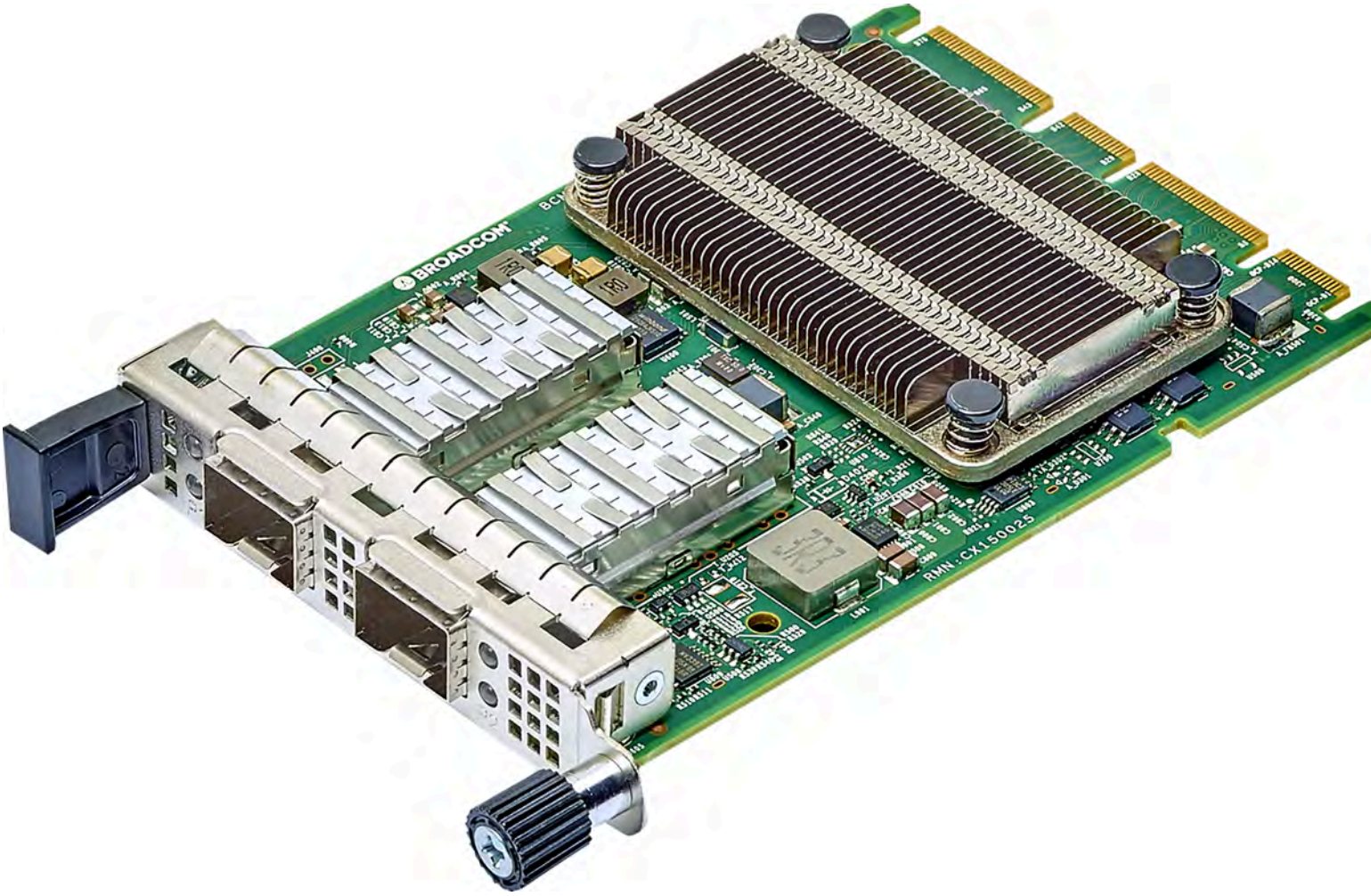
The information in the following table applies to the Broadcom BCM957414N4140C network adapter and other similar 2x25G network adapters such as the BCM957414-N225DI0S and BCM957414-N225GI0S.

Table 102: BCM957414N4140C Functional Descriptions

Function	Description
Speed	Dual-Port 25 Gb/s Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of G links transmit and receive traffic at the same time.
Interface	SFP28 for 25 Gb/s
Device	Broadcom BCM57414 25 Gb/s MAC controller with integrated dual-channel 25 Gb/s SFP28 transceiver.

Function	Description
NDIS Name	Broadcom NetXtreme-E Series Dual-Port 25Gb SFP28 Ethernet OCP 3.0 Adapter
UEFI Name	Broadcom NetXtreme-E Dual 25Gb SFP28 OCP 3.0 Ethernet

Figure 49: BCM957414N4140C Small-Form-Factor Network Adapter



Note: The previous figure shows the pull-tab faceplate installed.

BCM957412-P410TGP0 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957412-P410TGP0 network adapter and other similar 4x10G Base-T network adapters such as the BCM957412-P410TDF0S, BCM957412-P410TGF0S, BCM957412-P410TDL0S, and BCM957412-P410TGF0S.

Table 103: BCM957412-P410TGP0 Functional Descriptions

Function	Description
Speed	Quad-Port 10GBASE-T Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 25G links transmit and receive traffic at the same time.
Interface	RJ-45 for 10 Gb/s
Device	Two Broadcom BCM57412 10 Gb/s MAC controllers connected to two BCM84892 10GBASE-T PHYs.
NDIS Name	Broadcom Quad Port 10GBASE-T PCIe Ethernet NIC
UEFI Name	Broadcom Quad Port 10GBASE-T PCIe Ethernet NIC

Figure 50: BCM957412-P410TGP0 Network Interface Card

Note: The previous figure shows the standard-profile bracket installed.

BCM957412-N410TGP0 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957412-N410TGP0 network adapter and other similar 4x10G Base-T network adapters such as the BCM957412-N410TDI0S and BCM957412-N410TGI0S.

Table 104: BCM957412-N410TGP0 Functional Descriptions

Function	Description
Speed	Quad-Port 10GBASE-T Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 25G links transmit and receive traffic at the same time.
Interface	RJ-45 for 10 Gb/s
Device	Two Broadcom BCM57412 10 Gb/s MAC controllers connected to two BCM84892 10GBASE-T PHYs.
NDIS Name	Broadcom Quad Port 10GBASE-T OCP Ethernet NIC
UEFI Name	Broadcom Quad Port 10GBASE-T OCP Ethernet NIC

Figure 51: BCM957412-N410TGP0 Network Interface Card

Note: The previous figure shows the standard-profile bracket installed.

BCM957416A4160C Ethernet Network Adapter

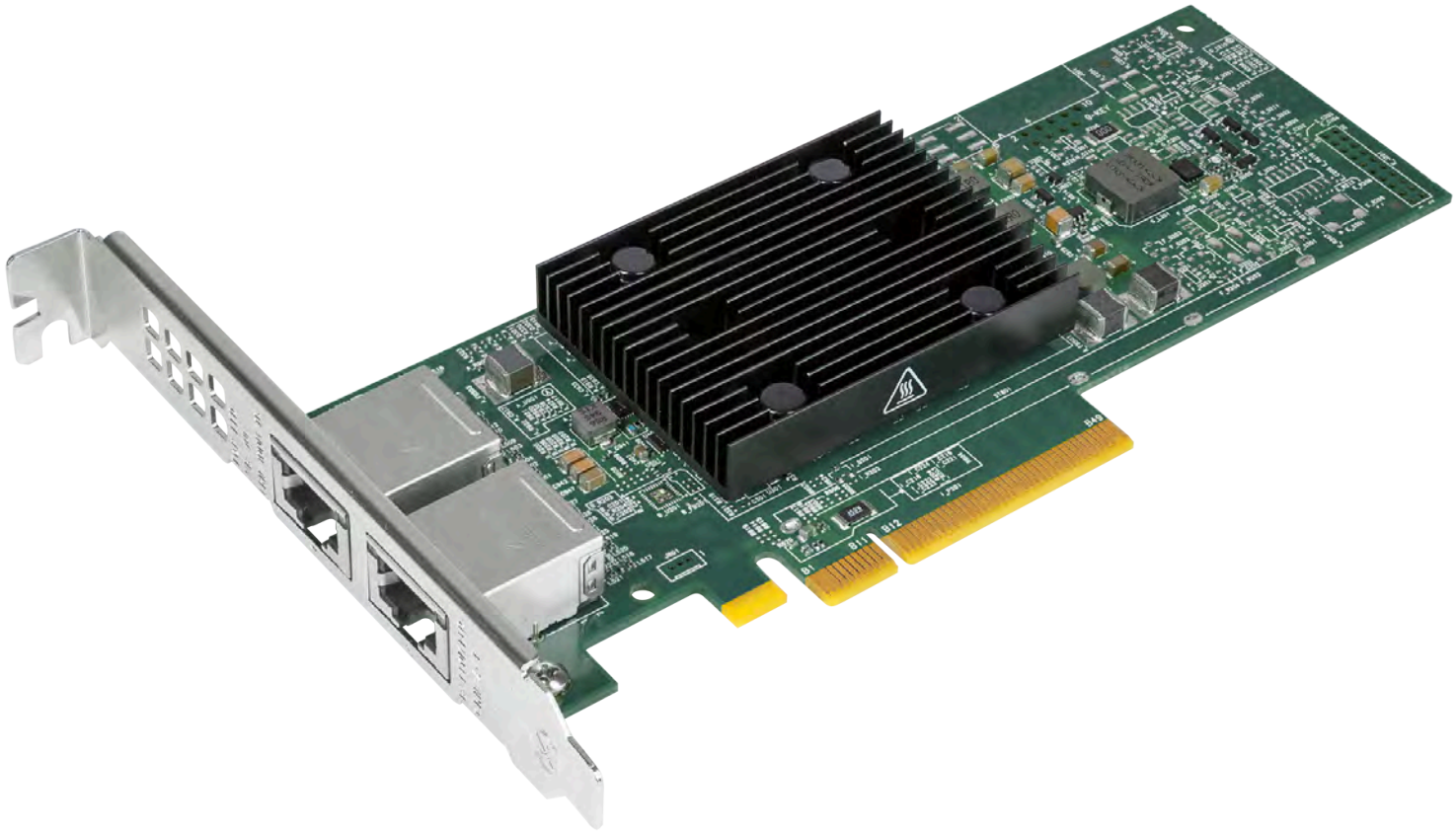
The information in the following table applies to the Broadcom BCM957416A4160C network adapter and other similar 2x10G Base-T network adapters such as the BCM957412-P210TDF0S, BCM957412-P210TGF0S, BCM957412-P210TDL0S, and BCM957412-P210TGF0S.

Table 105: BCM957416A4160C Functional Descriptions

Function	Description
Speed	Dual-Port 10GBASE-T Ethernet
PCIe	Gen3 x8

Function	Description
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 10G links transmit and receive traffic at the same time.
Interface	RJ-45 for 10 Gb/s
Device	Broadcom BCM57416 10 Gb/s MAC controller with integrated dual-channel 10GBASE-T transceiver.
NDIS Name	Broadcom NetXtreme-E Series Dual-Port 10GBASE-T Ethernet PCIe Adapter
UEFI Name	Broadcom Dual 10GBASE-T Ethernet

Figure 52: BCM957416A4160C Network Interface Card



Note: The previous figure shows the standard-profile bracket installed.

BCM957416N4160C Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957416N4160C network adapter and other similar 2x10G Base-T network adapters such as the BCM957412-N210TDI0S and BCM957412-N210TGI0S.

Table 106: BCM957416N4160C Functional Descriptions

Function	Description
Speed	Dual-Port 10GBASE-T Ethernet
PCIe	Gen3 x8
	The NIC supports PCIe Gen3, 2, and 1 speeds. However, PCIe Gen3 is recommended to achieve nominal throughput when 2 ports of 10G links transmit and receive traffic at the same time.
Interface	RJ-45 for 10 Gb/s
Device	Broadcom BCM57416 10 Gb/s MAC controller with integrated dual-channel 10GBASE-T transceiver.
NDIS Name	Broadcom NetXtreme-E Series Dual-Port 10GBASE-T Ethernet OCP 3.0 Adapter
UEFI Name	Broadcom NetXtreme-E 2Px10GBASE-T OCP 3.0 Ethernet

Figure 53: BCM957416N4160C Small-Form-Factor Network Adapter

Note: The previous figure shows the pull-tab faceplate installed.

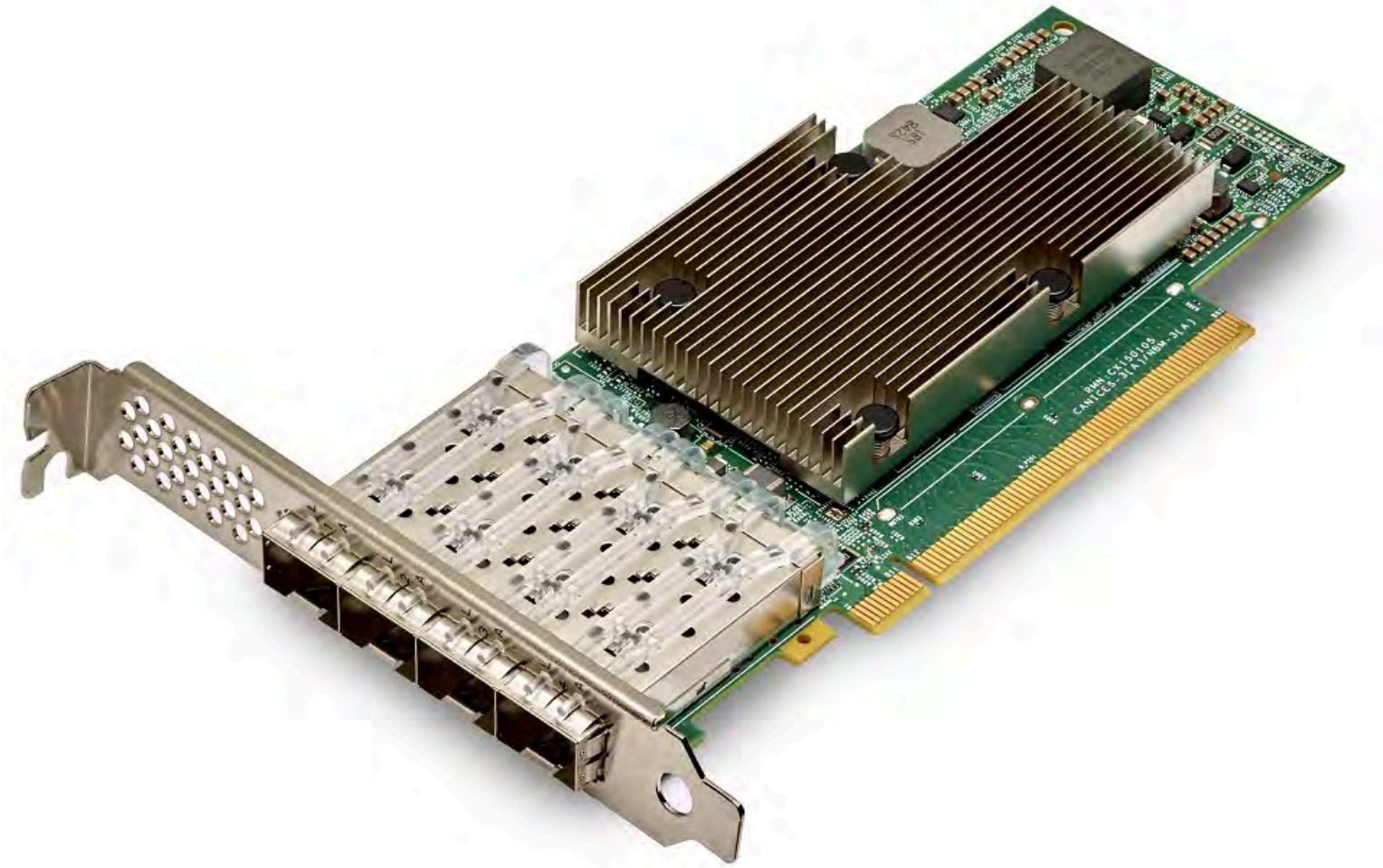
BCM957504-P425G Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957504-P425G network adapter.

Table 107: BCM957504-P425G Functional Descriptions

Function	Description
Speed	Quad-Port 25 Gb/s Ethernet
PCIe	Gen4 x16
Interface	SFP28 for 25 Gb/s
Device	Broadcom BCM57504 100G Gb/s MAC controller.
NDIS Name	Broadcom NetXtreme E-Series Quad-port 25Gb SFP28 PCIe Ethernet Adapter
UEFI Name	Broadcom NetXtreme-E Quad 25Gb SFP28 PCIe Ethernet

Figure 54: BCM957504-P425G Network Interface Card



BCM957504-N425G Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957504-N425G network adapter.

Table 108: BCM957504-N425G Functional Descriptions

Function	Description
Speed	Quad-Port 25 Gb/s Ethernet
PCIe	Gen4 x16
Interface	SFP28 for 25 Gb/s
Device	Broadcom BCM57504 100G Gb/s MAC controller.
NDIS Name	Broadcom NetXtreme E-Series Quad-port 25Gb SFP28 PCIe Ethernet Adapter
UEFI Name	Broadcom NetXtreme-E Quad 25Gb SFP28 PCIe Ethernet

Figure 55: BCM957504-N425G OCP 3.0 SFF Card

Note: The previous figure shows the pull-tab faceplate installed.

BCM957504-N1100G Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957504-N1100G network adapter.

Table 109: BCM957504-N1100G Functional Descriptions

Function	Description
Speed	Single-Port 100 Gb/s Ethernet
PCIe	Gen4 x16
Interface	QSFP56 for 100 Gb/s
Device	Broadcom BCM57504 100 Gb/s MAC controller.
NDIS Name	Broadcom NetXtreme-E Series Single Port 100Gb OCP 3.0 Ethernet Adapter
UEFI Name	Broadcom NetXtreme-E Single 100-Gb OCP 3.0 Ethernet

Figure 56: BCM957504-N1100G OCP 3.0 SFF Network Adapter

Note: The previous figure shows the pull-tab bracket installed by default.

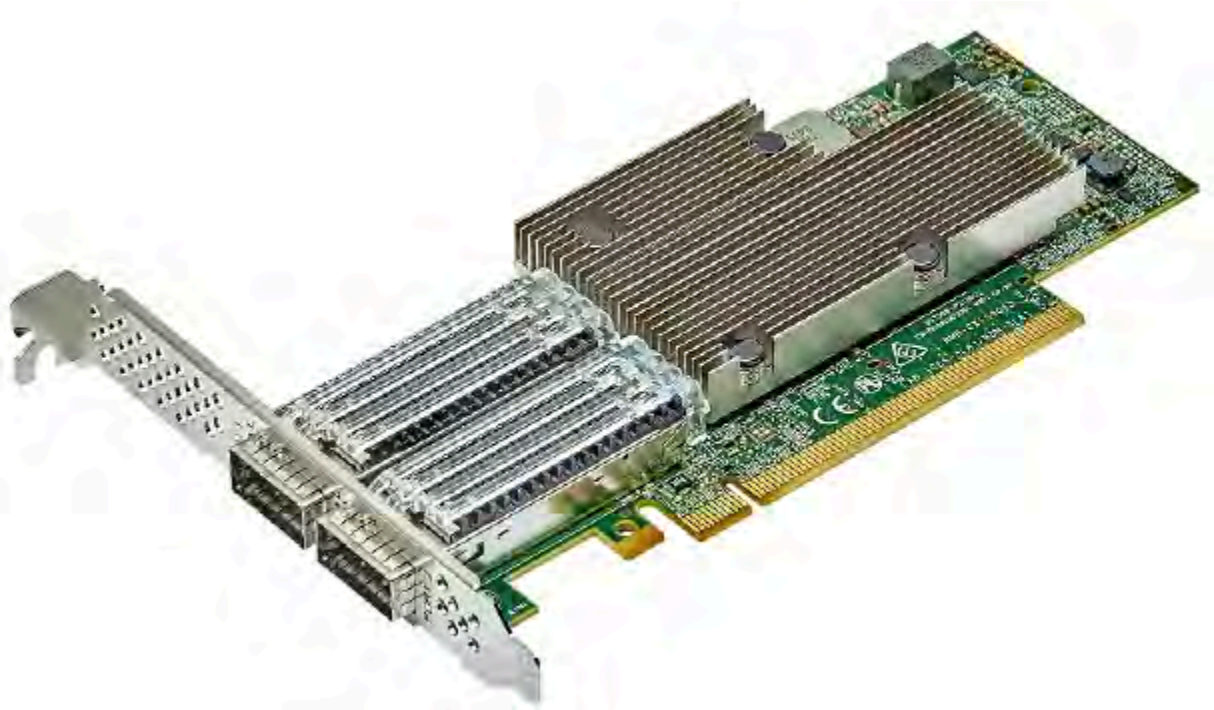
BCM957508-P2100G Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957508-P2100G network adapter.

Table 110: BCM957508-P2100G Functional Descriptions

Function	Description
Speed	Dual-Port 100 Gb/s Ethernet
PCIe	Gen4 x16
	The NIC supports PCIe Gen4, 3, 2, and 1 speeds. However, PCIe Gen4 is recommended to achieve nominal throughput when 2 ports of 100G links transmit and receive traffic at the same time.
Interface	QSFP56 for 100 Gb/s
Device	Broadcom BCM57508 200 Gb/s MAC controller.
NDIS Name	Broadcom P2100G NetXtreme-E Dual-Port 100Gb Ethernet PCIe Adapter
UEFI Name	Broadcom P2100G NetXtreme-E Dual 100Gb PCIe Ethernet

Figure 57: BCM957508-P2100G Network Interface Card



BCM957508-N2100G Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957508-N2100G network adapter.

Table 111: BCM957508-N2100G Functional Descriptions

Function	Description
Speed	Dual-Port 100 Gb/s Ethernet
PCIe	Gen4 x16
	The NIC supports PCIe Gen4, 3, 2, and 1 speeds. However, PCIe Gen4 is recommended to achieve nominal throughput when 2 ports of 100G links transmit and receive traffic at the same time.
Interface	QSFP56 for 100 Gb/s
Device	Broadcom BCM57508 200 Gb/s MAC controller.
NDIS Name	Broadcom N2100G NetXtreme-E Dual-Port 100Gb Ethernet PCIe Adapter
UEFI Name	Broadcom NetXtreme-E Dual 100Gb OCP 3.0 Ethernet

Figure 58: BCM957508-N2100G OCP 3.0 SFF Card

BCM957608-N425GP00 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957608-N425GP00 network adapter and other similar 4x25G network adapters such as the BCM957608-N425DSI00 and BCM957608-N425GSI00.

Table 112: BCM957608-N425GP00 Functional Descriptions

Function	Description
Speed	Quad-Port 25 Gb/s Ethernet
PCIe	Gen5 x16
Interface	SFP28 for 25 Gb/s
Device	Broadcom BCM57608 400 Gb/s MAC controller.
Device Name	Broadcom BCM957608-N425GP00 4x25G OCP3.0 NIC

Figure 59: BCM957608-N425GP00 Network Interface Card

BCM957608-P2200GQF00 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957608-P2200GQF00 network adapter and other similar 2x200G and 2x100G network adapters such as the BCM957608-P2100DQF00, BCM957608-P2100GQF20, BCM957608-P2100DQL00, BCM957608-P2100GQF20, BCM957608-P2200DQF00, BCM957608-P2200GQF20, BCM957608-P2200DQL00, and BCM957608-P2200GQF20.

Table 113: BCM957608-P2200GQF00 Functional Descriptions

Function	Description
Speed	Dual-Port 200 Gb/s and Single-Port 400 Gb/s Ethernet Dual-Port 100 Gb/s and Single-Port 200 Gb/s Ethernet
PCIe	Gen5 x16
Interface	QSFP112 for 200Gb/s
Device	Broadcom BCM57608 400 Gb/s MAC controller.
Device Name	Broadcom BCM957608-P2200 2x200G (2x100) PCIe Ethernet NIC

Figure 60: BCM957608-P2200GQF00 Network Interface Card

BCM957608-N2200GQP00 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957608-N2200GQP00 network adapter and other similar 2x200G and 2x100G network adapters such as the BCM957608-N2100DQI00, BCM957608-N2100GQI00, BCM957608-N2200DQI00, and BCM957608-N2200GQI00.

Table 114: BCM957608-N2200GQP00 Functional Descriptions

Function	Description
Speed	Dual-Port 200 Gb/s and Single-Port 400 Gb/s Ethernet Dual-Port 100 Gb/s and Single-Port 200 Gb/s Ethernet
PCIe	Gen5 x16
Interface	QSFP112 for 200Gb/s
Device	Broadcom BCM57608 400 Gb/s MAC controller.
Device Name	Broadcom BCM957608-P2200 2x200G (2x100) PCIe Ethernet NIC

Figure 61: BCM957608-N2200GQP00 Network Interface Card

BCM957608-P1400GDF00 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957608-P1400GDF00 network adapter.

Table 115: BCM957608-P1400GDF00 Functional Descriptions

Function	Description
Speed	Single-Port 400 Gb/s Ethernet
PCIe	Gen5 x16
Interface	QSFP-DD 112/QSFP-DD 56 for 400Gb/s
Device	Broadcom BCM57608 400 Gb/s MAC controller.
Device Name	Broadcom BCM957608-P1400 1x400G PCIe Ethernet NIC

Figure 62: BCM957608-P1400GDF00 Network Interface Card

BCM957608-N1400GDP00 Ethernet Network Adapter

The information in the following table applies to the Broadcom BCM957608-N1400GDP00 network adapter.

Table 116: BCM957608-N1400GDP00 Functional Descriptions

Function	Description
Speed	Single-Port 400 Gb/s Ethernet
PCIe	Gen5 x16
Interface	QSFP-DD 112/QSFP-DD 56for 400Gb/s
Device	Broadcom BCM57608 400 Gb/s MAC controller.
Device Name	Broadcom BCM957608-N1400 1x400G OCP Ethernet NIC

Figure 63: BCM957608-N1400GDP00 Network Interface Card

Ethernet Network Adapter Network Link and Activity LEDs

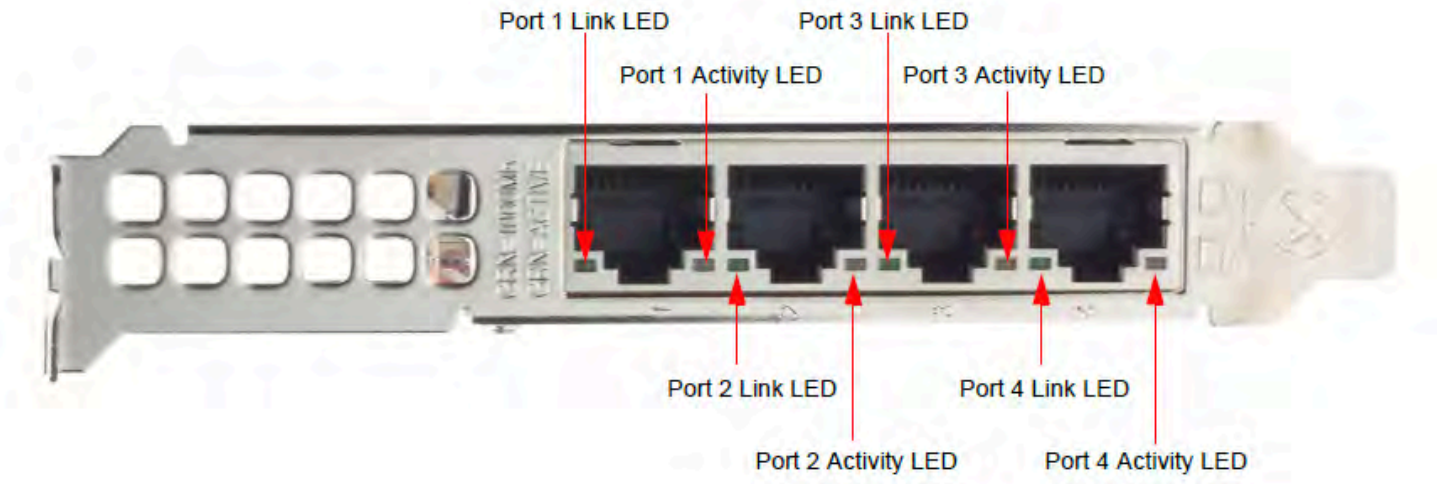
Provides information on the Ethernet connections, the state of the network link, and activity indicated by the LEDs on the rear connector.

Note: The following sections show the network interface cards in the BCM95719, BCM95720, BCM9574XX, BCM95750X, and BCM957608 families. The surface markings of the components and the bracket may not reflect the product upon receipt. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

BCM95719A1904AC Ethernet Network Adapter

The RJ-45 connector has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM95719A1904AC network adapter and other similar 4x1G Base-T network adapters such as the BCM95719-P41TDF0S, BCM95719-P41TDL0S, BCM95719-P41TGF0S, and BCM95719-P41TGL0S.

Figure 64: BCM95719A1904AC Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

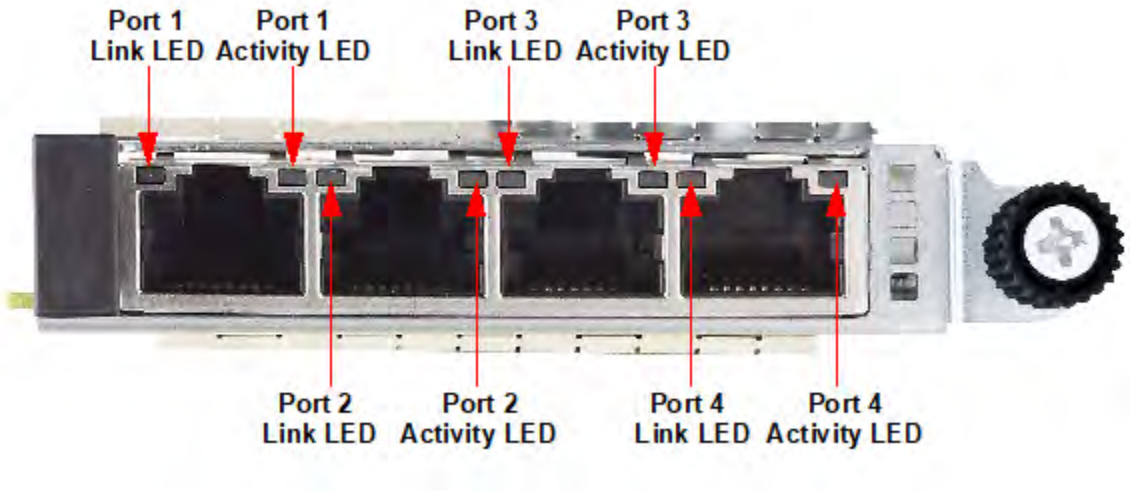
Table 117: BCM95719A1904AC Activity and Link LED Locations

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 1000 Mb/s
	Amber	Linked at lower speed

BCM95719N1905C Ethernet Network Adapter

The RJ-45 connector has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM95719N1905C network adapter and other similar 4x1G Base-T network adapters such as the BCM95719-N41TDI0S and BCM95719-N41TGI0S.

Figure 65: BCM95719N1905C Activity and Link LED Locations



Note: The previous figure shows the pull-tab faceplate installed. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

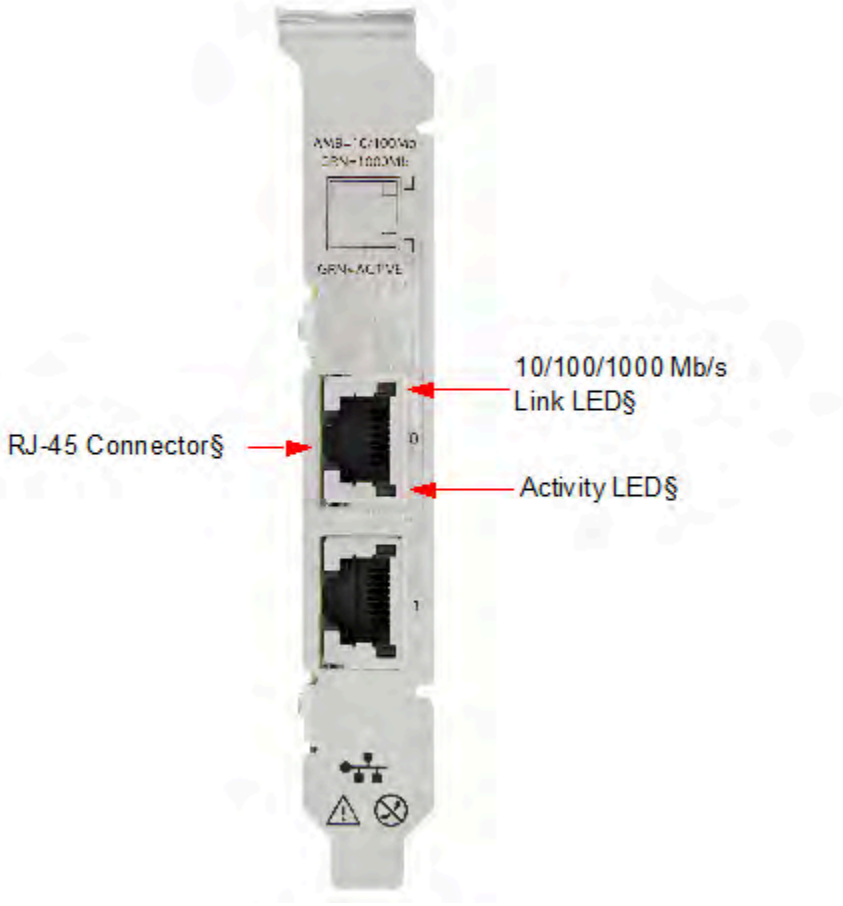
Table 118: BCM95719N1905C Activity and Link LED Locations

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 1000 Mb/s
	Amber	Linked at 10/100 Mb/s

BCM95720A2003AC Ethernet Network Adapter

The RJ-45 connector has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table.

Figure 66: BCM95720A2003AC Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

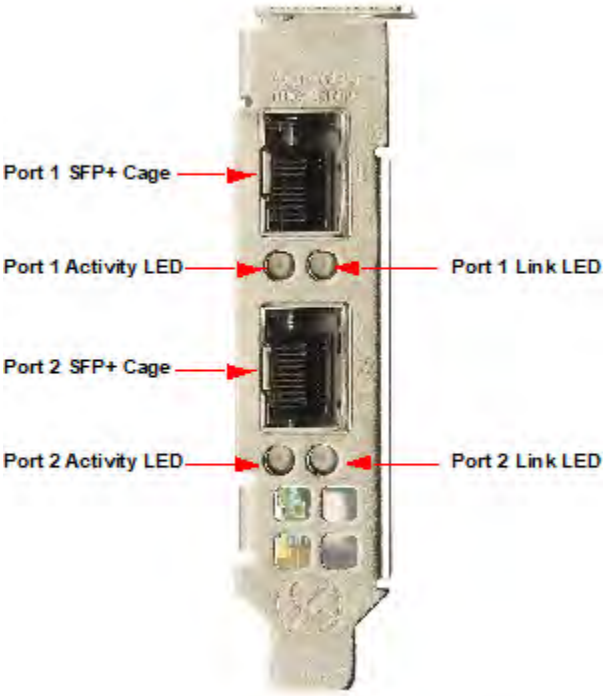
Table 119: BCM95720A2003AC Activity and Link LED Locations

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 1000 Mb/s
	Yellow	Linked at 10 Mb/s or 100 Mb/s

BCM957412A4120AC Ethernet Network Adapter

The SFP+ port has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table.

Figure 67: BCM957412A4120AC Activity and Link LED Locations



Note: The previous figure shows the low-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

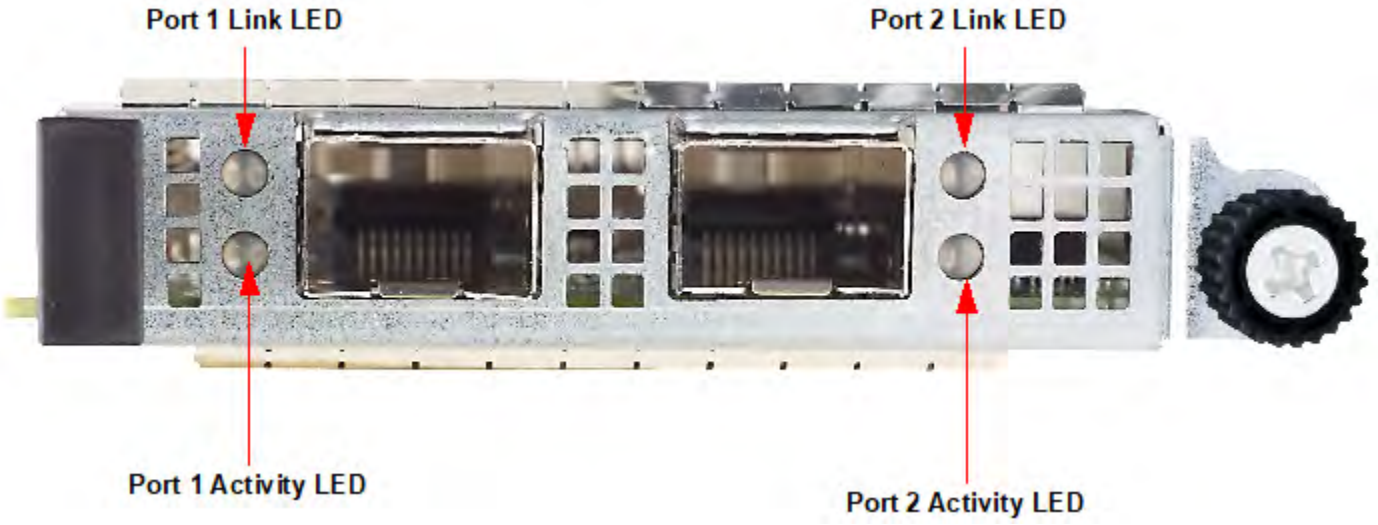
Table 120: BCM957412A4120AC Activity and Link LED Locations

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Yellow	Linked at 1 Gb/s

BCM957412N4120C Ethernet Network Adapter

The SFP+ port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible as shown in the following figure. Its locations and form factors conform to the OCP 3.0 Design Specification.

Figure 68: BCM957412N4120C Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed by default. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

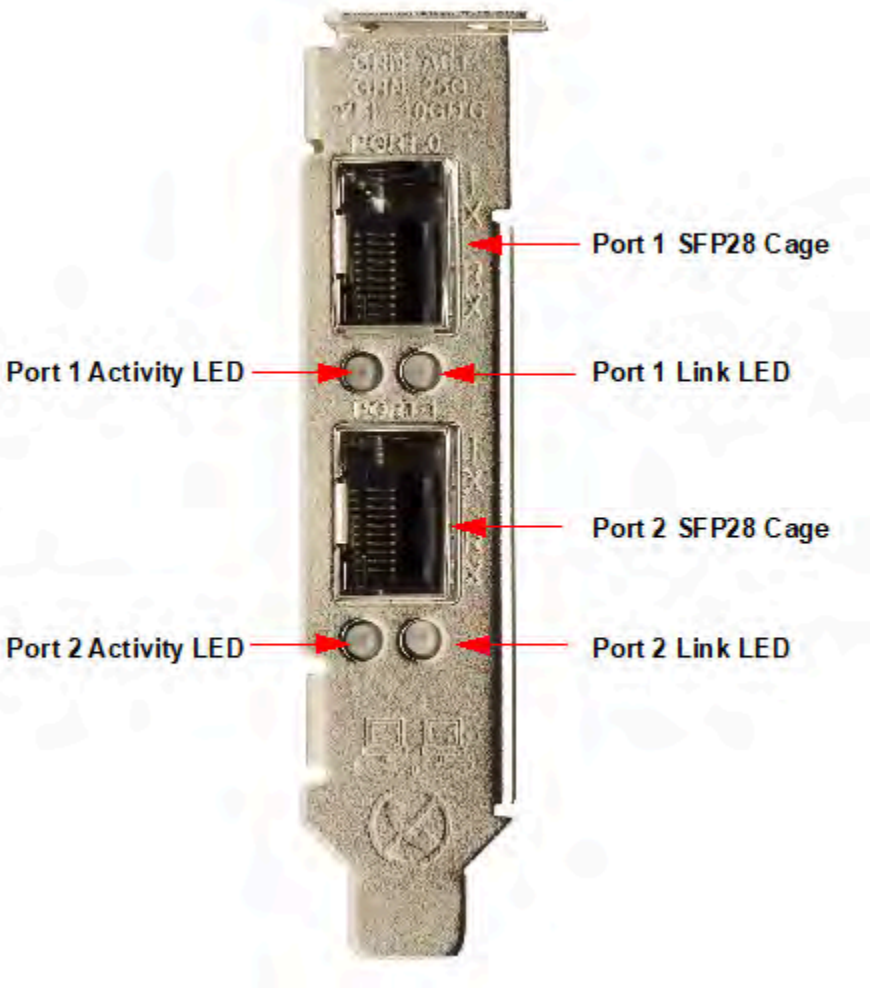
Table 121: BCM957412N4120C Network Adapter Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Amber	Linked at 1 Gb/s

BCM957414A4142CC Ethernet Network Adapter

The SFP28 port has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM957414A4142CC network adapter and other similar 2x25G network adapters such as the BCM957414-P225DF0S, BCM957414-P225GF0S, BCM957414-P225DL0S, and BCM957414-P225GF0S.

Figure 69: BCM957414A4142CC Activity and Link LED Locations



Note: The previous figure shows the low-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

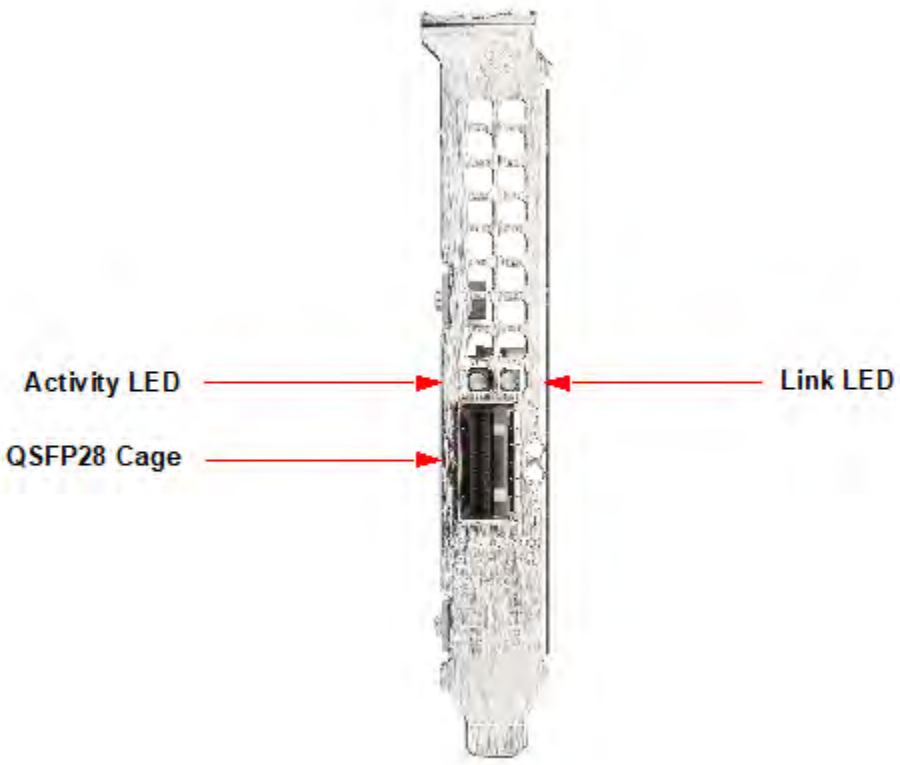
Table 122: BCM957414A4142CC Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 25 Gb/s
	Yellow	Linked at 1 Gb/s or 10 Gb/s

BCM957414A4140C Ethernet Network Adapter

The QSFP28 port has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM957414N4140C network adapter and other similar 2x25G network adapters such as the BCM957414-N225DI0S and BCM957414-N225GI0S.

Figure 70: BCM957414A4140C Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

Table 123: BCM957414A4140C Activity and Link LED Locations

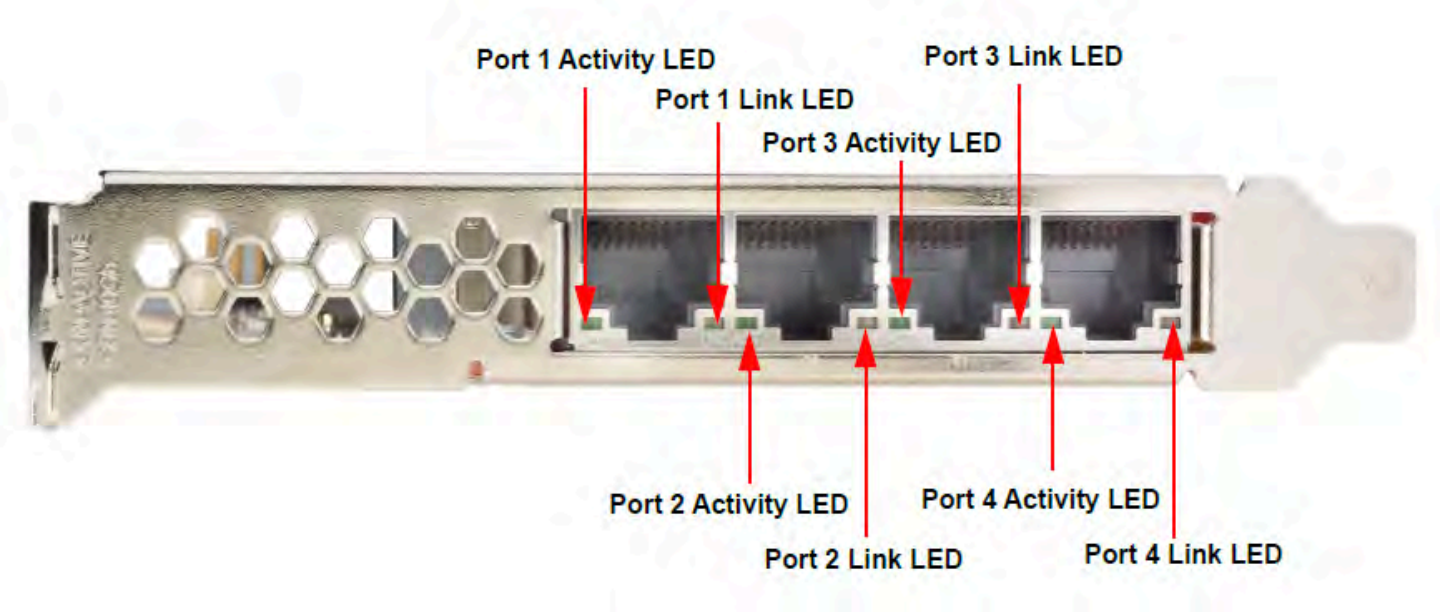
LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 50 Gb/s
	Yellow	Linked at lower speed

BCM957412-P410TGP0 Ethernet Network Adapter

Each Ethernet interface has a link LED to indicate link status and an activity LED to indicate data traffic. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom

BCM957412-P410TGP0 network adapter and other similar 4x10G Base-T network adapters such as the BCM957412-P410TDF0S, BCM957412-P410TGF0S, BCM957412- P410TDL0S, and BCM957412-P410TGF0S.

Figure 71: BCM957412-P410TGP0 Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

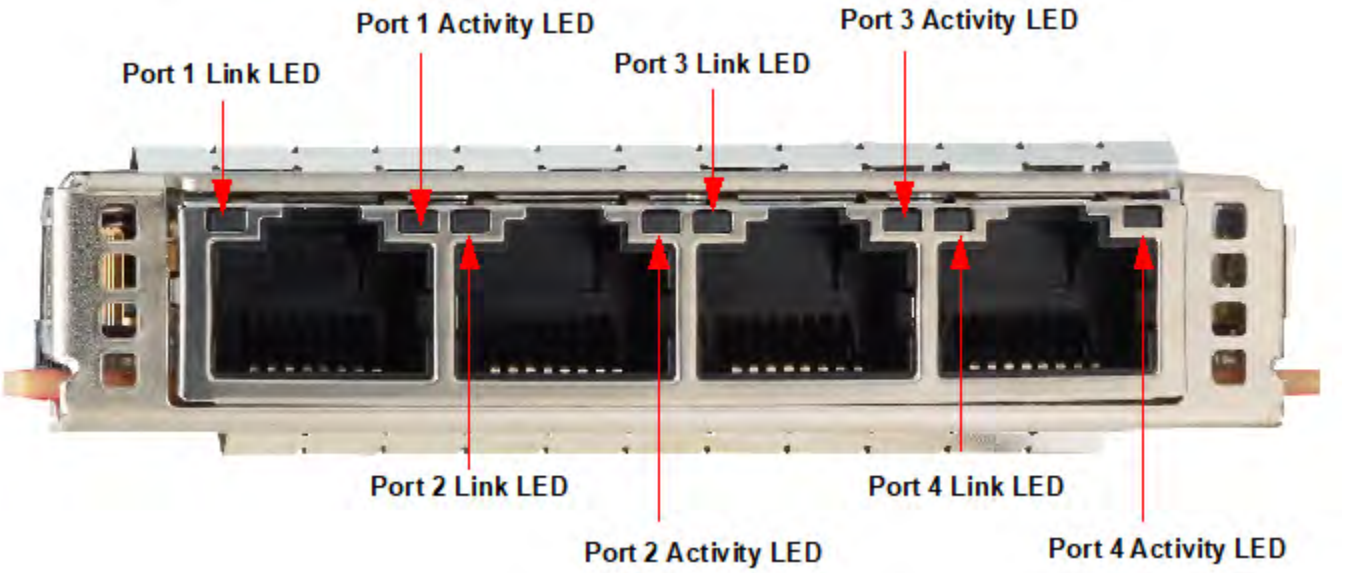
Table 124: BCM957412-P410TGP0 Activity and Link LED Functions

LED Type	Color/Behavior	Notes
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Amber	Linked at 1 Gb/s

BCM957412-N410TGP0 Ethernet Network Adapter

Each Ethernet interface has a link LED to indicate link status and an activity LED to indicate data traffic. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM957412-N410TGP0 network adapter and other similar 4x10G Base-T network adapters such as the BCM957412-N410TDI0S and BCM957412-N410TGI0S.

Figure 72: BCM957412-N410TGP0 Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed by default. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

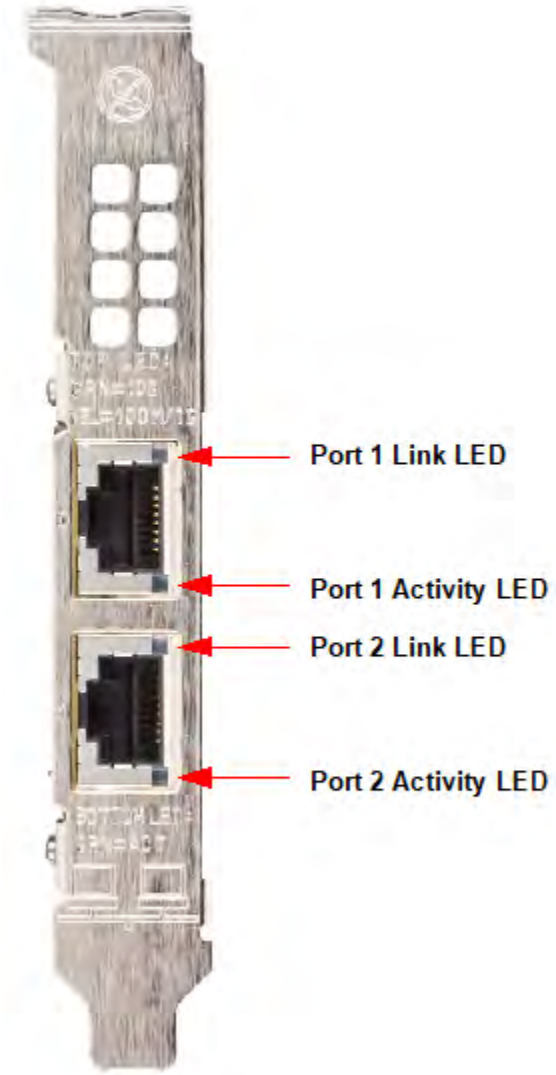
Table 125: BCM957412-N410TGP0 Activity and Link LED Functions

LED Type	Color/Behavior	Notes
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Amber	Linked at 1 Gb/s

BCM957416A4160C Ethernet Network Adapter

Each Ethernet interface has a link LED to indicate link status and an activity LED to indicate data traffic. The LEDs are shown in the following figure and described in the following table. This LED information applies to the Broadcom BCM957416A4160C network adapter and other similar 2x10G Base-T network adapters such as the BCM957412-P210TDF0S, BCM957412-P210TGF0S, BCM957412- P210TDL0S, and BCM957412-P210TGF0S.

Figure 73: BCM957416A4160C Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

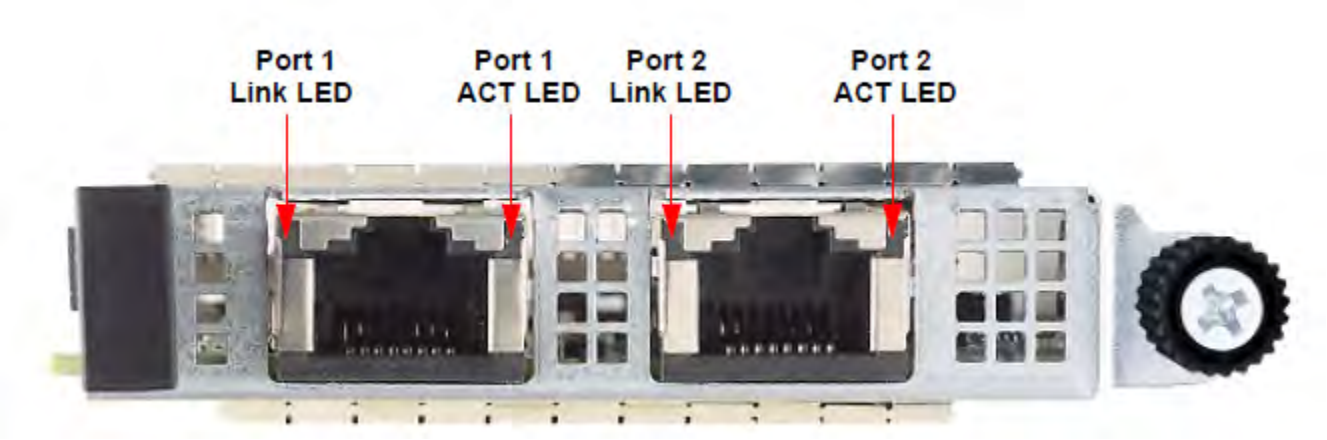
Table 126: BCM957412-P410TGP0 Activity and Link LED Functions

LED Type	Color/Behavior	Notes
Activity	Off	No Activity
	Green blinking	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Amber	Linked at 1 Gb/s

BCM957416N4160C Ethernet Network Adapter

Each Ethernet interface has a link LED to indicate link status and an activity LED to indicate data traffic. The LEDs are shown in the following figure and described in the following table.

Figure 74: BCM957416N4160C Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed by default. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

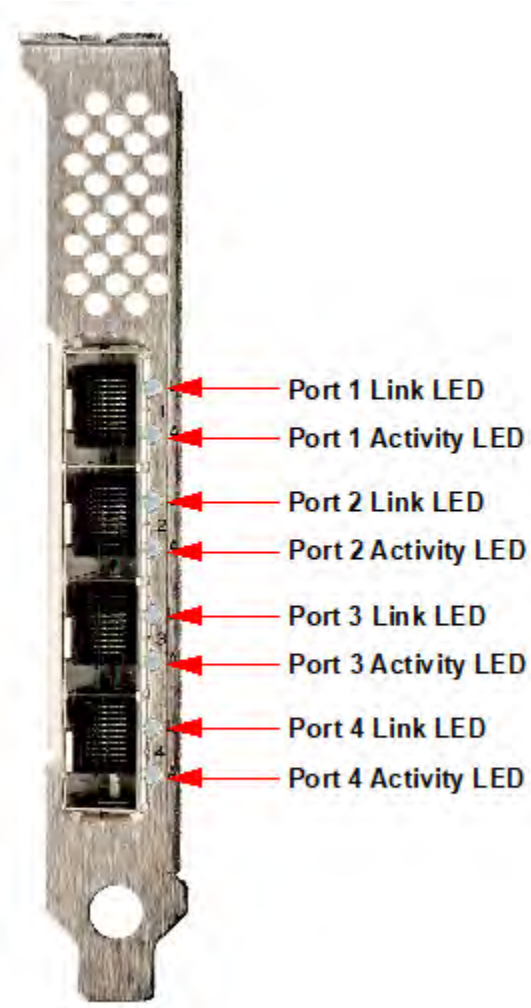
Table 127: BCM957416N4160C Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 10 Gb/s
	Amber	Linked at 1 Gb/s

BCM957504-P425G Ethernet Network Adapter

The SFP28 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure.

Figure 75: BCM957504-P425G Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

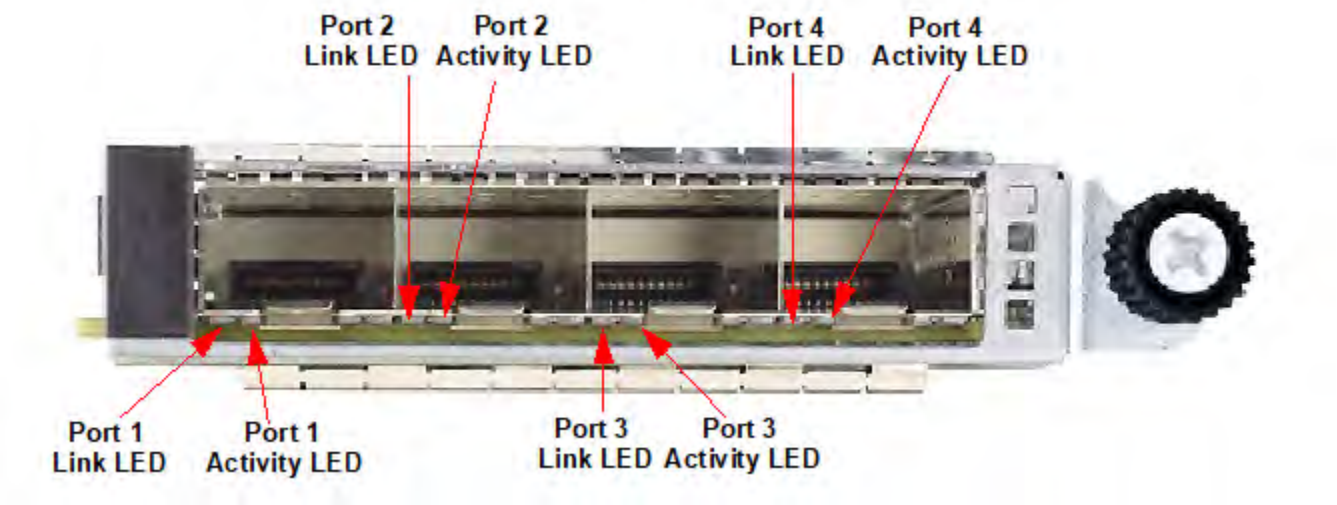
Table 128: BCM957504-P425G Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 25 Gb/s
	Amber	Linked at lower speed

BCM957504-N425G Ethernet Network Adapter

The SFP28 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible as shown in the following figure. Its locations and form factors conform to the OCP 3.0 Design Specification. The following table describes the LED functionality.

Figure 76: BCM957504-N425G Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed by default. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

Table 129: BCM957504-N425G Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 25 Gb/s
	Amber	Linked at lower speed

BCM957504-N1100G Ethernet Network Adapter

The QSFP56 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible as shown in the following figure. Its locations and form factors conform to the OCP 3.0 Design Specification. The following table describes the LED functionality.

Figure 77: BCM957504-N1100G Activity and Link LED Locations

Note: The previous shows the pull-tab bracket installed by default. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

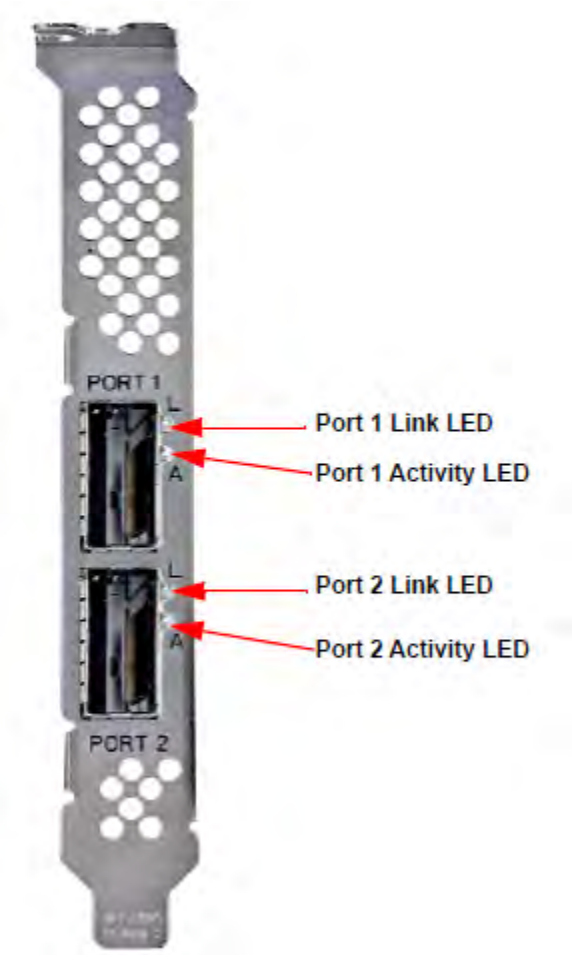
Table 130: BCM957504-N1100G Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 100 Gb/s
	Amber	Linked at lower speed

BCM957508-P2100G Ethernet Network Adapter

The QSFP56 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality.

Figure 78: BCM957508-P2100G Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

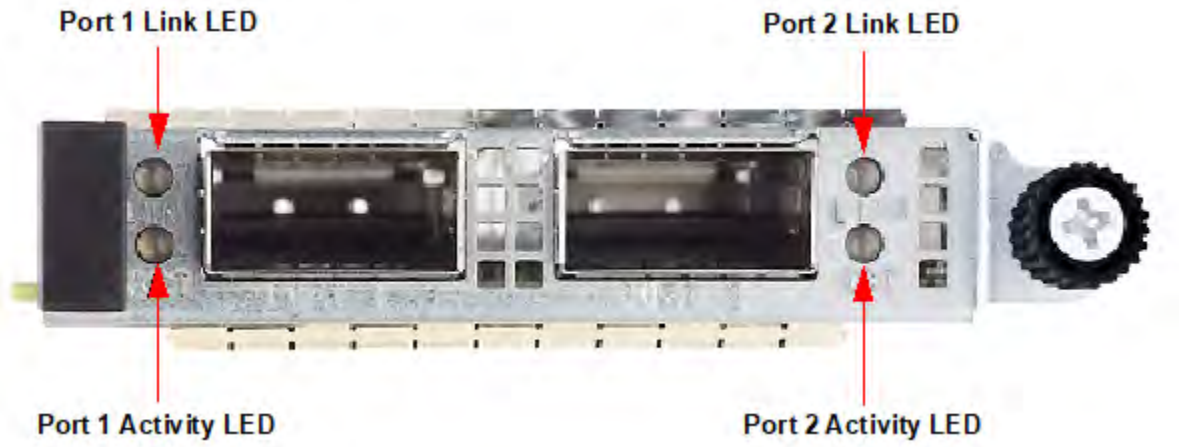
Table 131: BCM957508-P2100G Activity and Link LED Functions

LET Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Traffic Flowing Activity
Link	Off	No Link
	Green	Linked at 100 Gb/s
	Amber	Linked at lower speed

BCM957508-N2100G Ethernet Network Adapter

The QSFP56 port has two LEDs to indicate traffic activities and link speed. The LEDs are shown in the following figure and described in the following table. Its locations and form factors conform to the OCP 3.0 Design Specification.

Figure 79: BCM957508-N2100G Activity and Link LED Locations



Note: The previous figure shows the pull-tab faceplate installed. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

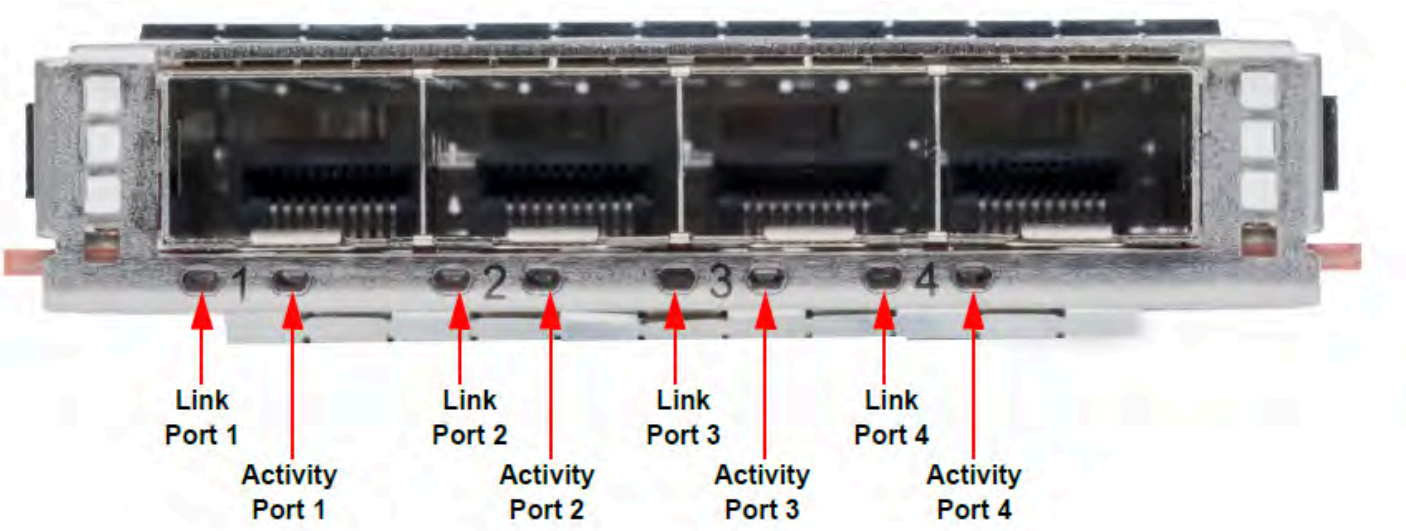
Table 132: BCM957508-N2100G Activity and Link LED Locations

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green blinking	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 100 Gb/s
	Amber	Linked at lower speed

BCM957608-N425GP00 Ethernet Network Adapter

The SFP port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality. This LED information applies to the Broadcom BCM957608-N425GP00 network adapter and other similar 4x25G network adapters such as the BCM957608-N425DSI00 and BCM957608-N425GSI00.

Figure 80: BCM957608-N425GP00 Activity and Link LED Locations



Note: The previous figure shows the pull-tab faceplate installed. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

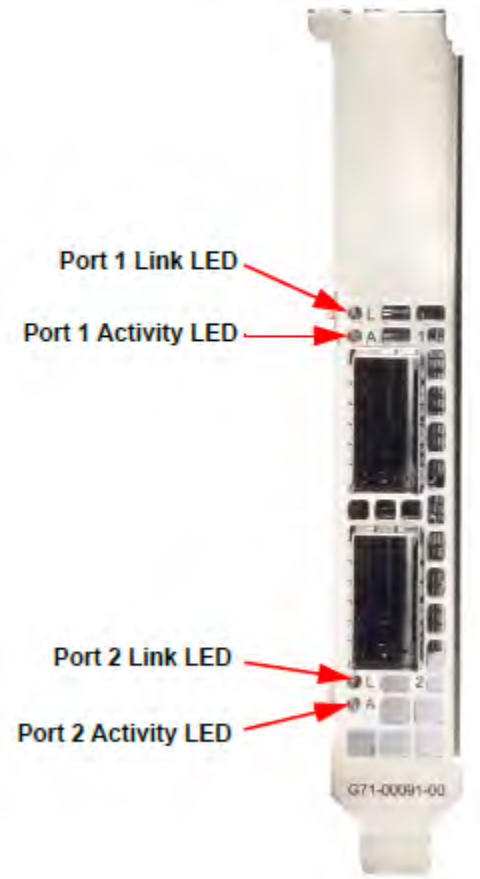
Table 133: BCM957608-N425GP00 Activity and Link LED Functions

LET Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 25 Gb/s
	Amber	Linked at lower speed

BCM957608-P2200GQF00 Ethernet Network Adapter

The QSFP112 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality. This LED information applies to the Broadcom BCM957608-P2200GQF00 network adapter and other similar 2x200G and 2x100G network adapters such as the BCM957608-P2100DQF00, BCM957608-P2100GQF20, BCM957608-P2100DQL00, BCM957608-P2100GQF20, BCM957608-P2200DQF00, BCM957608-P2200GQF20, BCM957608-P2200DQL00, and BCM957608-P2200GQF20.

Figure 81: BCM957608-P2200GQF00 Activity and Link LED Locations



Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

Note: If the 2 x 200G ports are aggregated together as a single 400G port, only Port 1 LEDs are active.

Table 134: BCM957608-P2200GQF00 Activity and Link LED Functions

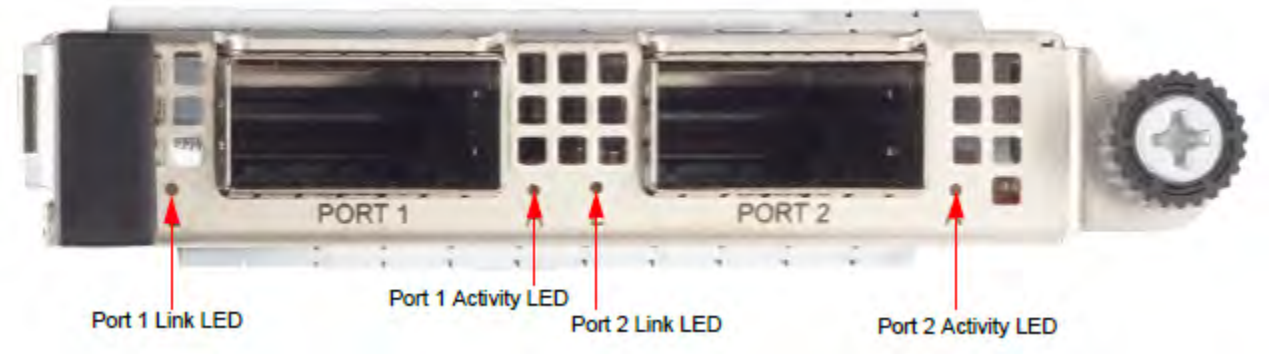
LET Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 200 Gb/s and 400 Gb/s
	Amber	Linked at lower speed

BCM957608-N2200GQP00 Ethernet Network Adapter

The QSFP112 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality. This LED information applies to the Broadcom BCM957608-N2200GQP00 network adapter and other similar 2x200G and 2x100G

network adapters such as the BCM957608-N2100DQI00, BCM957608-N2100GQI00, BCM957608-N2200DQI00, and BCM957608-N2200GQI00.

Figure 82: BCM957608-N2200GQP00 Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

Note: If the both ports are aggregated together as a single port, only Port 1 LEDs are active.

Table 135: BCM957608-N2200GQP00 Activity and Link LED Functions

LED Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 200 Gb/s and 400 Gb/s (N2200) Linked at 100 Gb/s and 200 Gb/s (N2100)
	Amber	Linked at lower speed

BCM957608-P1400GDF00 Ethernet Network Adapter

The QSFP-DD 112 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality.

Figure 83: BCM957608-P1400GDF00 Activity and Link LED Locations

Note: The previous figure shows the standard-profile bracket installed. The surface markings of the components might not reflect the product received. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

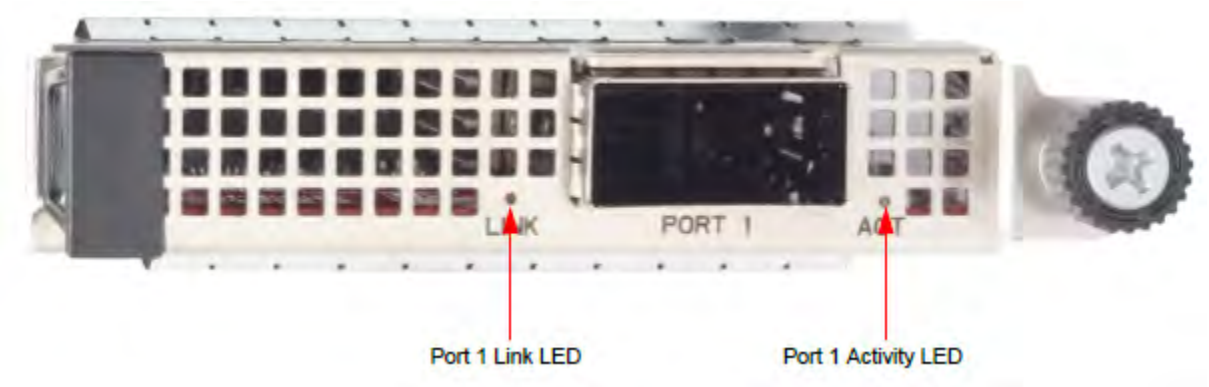
Table 136: BCM957608-P1400GDF00 Activity and Link LED Functions

LET Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 400 Gb/s
	Amber	Linked at lower speed

BCM957608-N1400GDP00 Ethernet Network Adapter

The QSFP-DD 112 port supports two LEDs to indicate traffic activities and link speed. The LEDs are visible through the cutout on the bracket as shown in the following figure. The following table describes the LED functionality.

Figure 84: BCM957608-N1400GDP00 Activity and Link LED Locations



Note: The previous figure shows the pull-tab bracket installed. The surface markings of the components might not reflect the product received. The position of the LEDs and port indicators are equivalent for the internal lock bracket. Broadcom reserves the right to change any component on the printed circuit board with the same functionality.

Table 137: BCM957608-N1400GDP00 Activity and Link LED Functions

LET Type	Color/Behavior	Note
Activity	Off	No Activity
	Green (blinking)	Link up (traffic flowing)
Link	Off	No Link
	Green	Linked at 400 Gb/s
	Amber	Linked at lower speed

Ethernet Network Adapter Regulatory and Safety Approvals

Details the regulatory and safety approvals for Ethernet network adapters. See the individual data sheets for the product classification and referenced standard dates.

Table 138: Regulatory Approvals

Standard/Country	Certification Type	Compliance
CE/EU	EN 55032 EN 55024/EN 55035 EN 61000-3-2 EN 61000-3-3	CE report and CE sDoC
UKCA/United Kingdom	EN 55032 EN 55024/EN 55035 EN 61000-3-2 EN 61000-3-3	UKCA DoC
FCC/USA	CFR47, Part 15	FCC sDoC and EMC report referencing FCC part 15 regulations
IC/Canada	ICES-003	Report referencing IC standards
ACA/Australia, New Zealand	AS/NZS CISPR 32	sDoC certificate RCM Mark
BSMI/Taiwan	CNS13438, CNS15663	BSMI certificate

MIC/S. Korea	KN 32 and KN 35	Korea certificate MSIP Mark
VCCI/Japan	V-3/VCCI CISPR 32	Copy of VCCI online certificate

Table 139: Safety Approvals

Item	Applicable Standard	Approval/Certificate
CE/European Union	IEC 62368-1	CB report and certificate
UL/USA	IEC 62368-1 CTUVus UL	UL report and certificate
CSA/Canada	CSA 22.2 No. 950	CSA report and certificate

Frequently Asked Questions for Ethernet Network Adapters

Provides resolutions to common questions and issues for Ethernet network adapters.

This section contains information on the following frequently asked questions.

Why does the system report TX timeouts on rings that are not configured for strict priority CoS using NICCLI?

Using strict priority for a given CoS can result in starvation of other configured rings. This may result in the inability to transmit packets on ETS rings and also result in the kernel reporting transmit timeouts. Configure strict priority only on rings with high-priority, low-throughput traffic to prevent consuming resources.

N-Tuple Filter Specific FAQs

This section provides n-tuple filter specific frequently asked questions and answers.

Ethtool filter creation with UDP protocol and dport 4789 and 4791 fails.

UDP destination port 4789 and 4791 are standard VXLAN and RoCEv2 ports that are offloaded by the Ethernet adapter. N-tuple filters using these UDP ports conflict with the offload features of the Ethernet adapter which is why these filters cannot be created.

Ethtool N tuple filters that are created with port number 0 will not operate.

The Internet Assigned Numbers Authority (IANA) maintains an official listing of the intended usage of these port numbers on the Internet, and system port 0 cannot be used.

A hash collision may occur while creating a maximum of 5 tuple filters on the Ethernet adapter.

The exact match matching is based on a hash calculation. Mathematically, it is possible that 5 tuples can produce the same hash value as another 5 tuples using the same hash algorithm. The odds of a collision increase as the number of bits used to represent a hash value is reduced.

Ethtool ntuple filters that are created with known ports like RoCE, VXLAN, and so forth, will not operate.

The RoCE and Vxlan ports are not available for general socket applications. Users of ethtool n tuple filter should not create filters with RoCE and VXLAN ports since no socket applications are using these ports. ntuple filters are now supported on stacked interfaces such as VLAN, bond, and Linux bridge interfaces.

Parameters `disable_gre_ver_check` , `disable_nvgre_rx` , and `disable_nvgre_tx` must be set using NICCLI on the Ethernet adapter for flow steering to be functional on GRE packets.

These settings are required to properly enable GREv1 support. For users who are interested in GREv1 flow steering and stateless offloads such as TPA, TSO, and RSS, these settings should be configured using the NICCLI tool.

What is the maximum number of N tuple filters that are supported on Broadcom Ethernet Adapters?

It is difficult to estimate the maximum number of N tuple filters without knowing the system configuration and environment. This number varies and can only be provided on a case-by-case basis. The best way to do this is to provide a query feature in the Broadcom tool to display the maximum N tuple filters in a particular system.

The ethtool -n <iface name> does not list n tuple filters in the order in which they are created on Ethernet Adapters.

ethtool displays the n tuple filters in the order that they are stored internally in the driver's data structures. The order is different from the order in which they were created.

The Ethernet adapter shows "Device Reset is not supported on this device" after completing a hot reset in a loop of 1654 iterations.

The Ethernet adapter may show "Device Reset is not supported on this device" after performing a hot reset in a loop for more than 1500 iterations. This is a known issue.

RoCE Specific FAQs

This section provides RoCE-specific frequently asked questions and answers.

Why `ib_read_bw` /`ib_write_bw` fails when running with high-QP count (>= 32 K) and low-link speed (<= 25 Gbps)

When the `ib_read_bw` or `ib_write_bw` test is run with high-QP count and low-link speed, it can fail with status 12 (`IBV_WC_RETRY_EXC_ERR`):

```
#bytes     #iterations    BW peak[MB/sec]    BW average[MB/sec]   MsgRate[Mpps]
Completion with error at client
Failed status 12: wr_id 349 syndrom 0xa
scnt=1827300, ccnt=1315300
Failed to complete run_iter_bw function successfully
```

This indicates that the max attempts at retransmission have been exhausted with the current ACK timeout. To correct, a higher ACK timeout value is recommended, which corresponds to a codepoint 21 (=8.58 sec) or more. In other words, add option `-u 21` in the corresponding command line.

Why dmesg reports timeout errors when a VM assigned a VF running RoCE traffic is powered off.

When a VM assigned a VF with active RoCE traffic running simultaneously on the VF and other functions (Pfs/VFs) is abruptly powered off, the following messages are seen:

```
[77178.374258] bnxt_en 0000:33:00.0 (unnamed net_device) (uninitialized): Error (timeout: 500015) msg {0x0
 0x0} len:0
[77180.870087] bnxt_en 0000:32:00.0 eno3np0: Error (timeout: 2000015) msg {0x23 0x1d94} len:0
[77184.269980] bnxt_en 0000:32:00.0 eno3np0: Error (timeout: 2000015) msg {0xb4 0x1d95} len:0
```

The presence of active RoCE traffic on the VF undergoing a Function Level Reset (FLR) or on any other PFs/VFs impacts the function initialization time of the VF undergoing FLR. Function initialization time scales linearly as the cumulative active QP count across all PFs and VFs increases. The increased function initialization time may lead to VF probe failures and periodic HWRM timeouts when the cumulative active QP count is greater than 6K QPs.

Why dmesg reports "Alloc doorbell page error" when the number of user processes scales up?

When more user applications are simultaneously active on a system, the following messages are seen:

```
[74480.019805] infiniband bnxt_re0: push dp alloc failed
[74480.020169] infiniband bnxt_re0: Alloc doorbell page failed!
```

This indicates that the system ran out of Doorbell pages for the new user application instance. To resolve this issue, increase the BAR 2 size using the NICCLI tool.

To get the current BAR 2 size using NICCLI for port 0, use the following command:

```
niccli -i 1 nvm --getoption pf_pci_bar2_size --scope 0

--------------------------------------------------------------------------------
Scrutiny NIC CLI v228.0.132.0 - Broadcom Inc. (c) 2023 (Bld-61.52.25.90.16.0)
--------------------------------------------------------------------------------

pf_pci_bar2_size = 16M
```

To set the BAR size as 128M on port 0, use the following command:

```
niccli -i 1 nvm --setoption pf_pci_bar2_size --scope 0 --value 128M

--------------------------------------------------------------------------------
Scrutiny NIC CLI v228.0.132.0 - Broadcom Inc. (c) 2023 (Bld-61.52.25.90.16.0)
--------------------------------------------------------------------------------

pf_pci_bar2_size is set successfully
```

Perform a cold boot on the system to apply the configuration changes.

The libibverbs: Warning: Driver bnxt_re does not support the kernel ABI message displays.

There are two possible resolutions:

- This can happen if the out-of-box `libbnxt_re` library has not been installed or the rdma-core package on the host has been updated which reinstalls inbox `libbnxt_re` . To resolve the error, rebuild and reinstall out-of-box `libbnxt_re` .
- This message occurs after a kernel upgrade or Linux distribution update. To resolve this issue, rebuild the bnxt_en and bnxt_re driver files with the newer kernel.

Why does the `-x 3` command parameter fail when used with ib perf tools?

GID[3] is required for `-x 3` to function. An `ifconfig <index> down; ifconfig <index> up` command is required to populate GID[3].

GIDs corresponding to IPv4 and IPv6 addresses may be missing after device creation sequences such as driver load or device error recovery.

Example: When RoCE v1 and RoCE v2 are enabled on the adapter, `ibv_devinfo -d <device> -vvv` shows:

```
GID[ 0]:
fe80:0000:0000:0000:5e6f:69ff:fe1e:2f3e, RoCE v1
GID[ 1]: fe80::5e6f:69ff:fe1e:2f3e, RoCE v2
```

The output should be as follows:

```
GID[ 0]: fe80:0000:0000:0000:5e6f:69ff:fe1e:2f3e, RoCE v1
GID[ 1]: fe80::5e6f:69ff:fe1e:2f3e, RoCE v2
GID[ 2]: 0000:0000:0000:0000:0000:ffff:c0a8:0033, RoCE v1
GID[ 3]: ::ffff:192.168.0.51, RoCE v2
GID[ 4]: 2001:0000:0000:0000:0000:0000:0000:0051, RoCE v1
GID[ 5]: 2001::51, RoCE v2
```

This is due to the device creation sequence from the netdev event context. The design change to avoid these failures will be available in future releases. As a workaround, bring down the L2 interface (`ifconfig down`) and then bring it up (`ifconfg up`). This forces the stack to add the GIDs again.

Is there a command to check PFC statistics?

The ethtool -s <InterfaceIndex> |grep pfc |more command shows the PFC statistics.

What happens to applications when migrating from an older Broadcom Release not using RoCEv1 to a new Broadcom Release that supports RoCEv1 and RoCEv2?

This example shows the `ibv_device` output from 2.23 release where RoCEv1 and RoCEv2 are both supported. Note the number of GIDs in the example. There are four GIDs composed of the following::

1. IPv6/RoCEv1 (GID index 0)

2. IPv6/RoCEv2 (GID index 1)
3. IPv4/RoCEv1 (GID index 2)
4. IPv4/RoCEv2 (GID index 3)

The IPv4 address is just an IPv4 address mapped into the IPv6 address space. This can be identified by 80 “0” bits, followed by 16 “1” bits (“FFFF” in hexadecimal), followed by the original 32-bit IPv4 address

```
ibv_devinfo -v -d bnxt_re0 | grep GID
ibv_devinfo -v -d bnxt_re0 | grep GID
GID[ 0]: fe80:0000:0000:0000:0205:06ff:fe03:0200
GID[ 1]: fe80:0000:0000:0000:0205:06ff:fe03:0200
GID[ 2]: 0000:0000:0000:0000:0000:ffff:c9c9:c90a
GID[ 3]: 0000:0000:0000:0000:0000:ffff:c9c9:c90a

cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/0

IB/RoCE v1
cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/1
RoCE v2
cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/2
IB/RoCE v1
cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/3
RoCE v2
```

The following example shows the ibv_device output from a Broadcom release (2.19-2.22) where only RoCEv2 is supported. Note the number of GIDs. There are 2 GIDs:

1. IPv6/RoCEv2 (GID index 0)
2. IPv4/RoCEv2 (GID index 1)

```
ibv_devinfo -v -d bnxt_re0 | grep GID
GID[ 0]: fe80:0000:0000:0000:0205:06ff:fe03:0200
GID[ 1]: 0000:0000:0000:0000:0000:ffff:c9c9:c90a
cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/0
RoCE v2
cat /sys/class/infiniband/bnxt_re0/ports/1/gid_attrs/types/1
RoCE v2
```

As part of migrating the application, the correct GID must be used. When using applications that need the GID to be specified as a parameter, ensure that the correct GID is used after any of the following link events:

- When using perftools, -x is used as the index into the array of GIDs support.

Why do atomic operations failover peer devices such as GPUs?

To support IB atomic operations (like `ib_atomic_bw`) over peer devices like GPUs, the device should advertise Atomic Operation completion capability. To confirm if this capability is enabled for the GPU, check the PCI express capability for AtomicOpsCap using `lspci -s <b:d.f> -vvv` .

Attempting atomic operations over a peer device that does not support atomic operations can cause failures due to PCI errors seen by Broadcom Ethernet adapters.

Why is a Resource unavailable for TX rings message received?

The following message is received:

```
2378.874679] bnxt_en 0000:32:00.0 eno12399np0: Resources unavailable for 24 tx rings, maximum 21 available
[ 2378.874684] infiniband bnxt_re0: Fail to setets rc:-12
```

```
[ 2378.874687] infiniband bnxt_re0: Fail to initialize Flow control
```

Some of the values in the previous message may vary, however, this indicates that 2 TCs for RoCE could not be configured when the driver was loaded. To use RoCE with PFC enabled, reduce the number of rings allocated to L2 using `ethtool -L`

Why are out of order TX/RX RoCE packets of the same RoCE connection displayed in the tcpdump or the wireshark output when capturing RoCE offload traffic on an ethernet device?

Both captured TX and RX RoCE offload traffic is routed to the L2 RX datapath, which is subjected to the RSS processing by default. Because of the difference in the UDP source port value between the captured TX and RX RoCE offload packets of the same RoCE connection, the TX and the RX RoCE offload packets are placed onto two different RX rings. Depending on the host processing, it is possible that the captured TX and RX RoCE offload packets can be received out of order by the host application such as tcpdump or wireshark. Before starting the capture, use **one** of the following options:

- Temporarily disable 4-tuple RSS hash for UDP so RSS hash is only performed on the source IP and destination IP addresses. When the capture is finished, restore the original setting.
  a. Query the existing UDP RSS setting and make note of current UDP RSS setting:`ethtool -u eth0 rx-flow-hash udp4`.
  b. Change the current UDP RSS setting to 2-tuple if the current UDP RSS setting is 4-tuple: `ethtool -U eth0 rx-flow-hash udp4 sd`.
  c. After the capture is complete, restore the original UDP RSS setting if the original setting is 4-tuple:`ethtool -U eth0 rx-flow-hash udp4 sdfn`.
- Temporarily change the number of RX rings to 1. When the capture is complete, restore the original setting.
  a. Show the current number of channels and make note of the current setting: `ethtool -l eth0`.
  b. Change the current channel setting to 1: `ethtool -L eth0 tx 0 rx 0 combined 1`.
  c. After the capture is complete, restore the original setting. For example, if the original number of channels is 16, issue `ethtool -L eth0 tx 0 rx 0 combined 16`.

RoCE LAG: Uneven QP distribution observed on LAG interfaces upon creating max QPs

The current implementation achieves load balancing on a per DPI (application) basis. If there are 100 applications creating 1 QP each, all QPs are created on the same port. Similarly, if there are 100 applications each creating an odd number of QPs, the QP count difference between the ports can be up to 100. Only when all the applications are creating an even number of QPs does the firmware guarantee that the difference in QP count between both ports is <= 2.

Need to capture RoCE packets via TCPDUMP using a BCM957608 device on a Linux server.

To capture RoCE packets via TCPDUMP:

1. Disable SR-IOV using the following command:
   ```
   niccli -i <index> nvm --setoption enable_sriov --value 0
   ```
2. Disable default EVB mode using the following command:
   ```
   niccli -i <index> nvm --setoption default_evb_mode --scope 0 --value 3
   ```
3. Reboot the server for the configuration options to take effect.

Note: These settings are persistent across reboots once they are configured via the NVM configuration options.

Examples:

```
#rdma link show  -  will show active device
link rocep10s0/1 state ACTIVE physical_state LINK_UP netdev enp10s0np0

Run tcpdump on the active interface
#tcpdump -i enp10s0np0
12:45:07.577815 ARP, Request who-has _gateway tell brcm1-PowerEdge-R760, length 28
```

```
12:45:07.577837 ARP, Request who-has _gateway tell brcm1-PowerEdge-R760, length 46
12:45:08.601885 ARP, Request who-has _gateway tell brcm1-PowerEdge-R760, length 28
12:45:08.601892 ARP, Request who-has _gateway tell brcm1-PowerEdge-R760, length 46
```

RoCE GID Change

Why are inconsistent RoCE V2 GID Indexes observed on some NIC interfaces in a cluster setup that have multiple nodes with multiple NICs?

The GID (Global Identifier) Index change normally occurs in a scenario when:

- There is an active QP.
- IP address is changed or the IP is removed and then re-added (most frequently, an abrupt IP address change would be triggered by a link flap).

Depending on the operating system distro and the network settings, following sequence of events might happen during link flap:

- Link goes down → IP address is removed → GID index is marked for deletion, but due to the active QP, the GID is not deleted from the table in the RoCE device driver and is marked pending for deletion.
- Link comes back up (shortly after going down, for example, it is link flap) → IP address is re-assigned → GID index is added for this new IP address. Since the prior GID index is still not deleted from the table (active QP holds the reference), a new GID index is assigned to the GID table entry for the new IP address.

Note: The refresh of the GID table is a kernel RoCE stack behaviour and not dependent on the NIC.

Examples:

- The GID table has indices 0-3 where Index 3 refers to IPv4 RoCEv2.
- Active QP uses IPv4 RoCEv2 so it is using GID index 3.
- Link goes down → IP address is removed, but active QP still holds GID index 3, which is marked pending for deletion.
- Link comes back up (while GID index 3 is still pending for deletion) → IP address is re-added → GIDs are re-scanned → GID 3 pending for deletion is skipped (since the QP still holds a reference) → new GID index (GID=4) is added.
- Instead of seeing GIDs 0, 1, 2, 3 (GID=3 being IPv4 RoCEv2) in the table, the indices are now 0, 1, 2, 4 (in other words, the kernel RoCE stack assigns GID=4 to IPv4 RoCEv2).

Impact

The GID change will impact running RoCE traffic where the GID index is hard-coded in applications that use socket based Connection Management (CM). Whether it is a performance test with `ib_write_bw` or AI/ML RCCL test across several nodes in a cluster setup, the change of GID index can result in failures.

How to Detect GID Change

Methods to monitor the GID change are as follows:

- Monitor the GID values for the port by using `ibv_devinfo` : This prints out the GIDs that are used for this port. For example:

```
$ ibv_devinfo -vvv -d bnxt_re0 | grep GID
GID[  0]:   fe80:0000:0000:0000:d604:e6ff:fe7c:4cb4, RoCE v1
GID[  1]:   fe80::d604:e6ff:fe7c:4cb4, RoCE v2
GID[  2]:   0000:0000:0000:0000:0000:ffff:c0a8:01ef, RoCE v1
GID[  4]:   ::ffff:192.168.1.239, RoCE v2
```

- Monitor async events using `ibv_asyncwatch` . For example:

```
$ ibv_asyncwatch -d bnxt_re0
bnxt_re0: async event FD 4
event_type IBV_EVENT_PORT_ERR (10), port 1
```

```
event_type IBV_EVENT_PORT_ACTIVE (9), port 1
event_type IBV_EVENT_GID_CHANGE (18), port 1
event_type IBV_EVENT_GID_CHANGE (18), port 1
…
```

How to Recover the Initial GID Index Table

1. Release the QP(s) that holds the GID pending for deletion (usually means stopping or killing the application, whether it is `ib_write_bw` or RCCL). Note that this step is application-dependent (in terms of how the user gracefully stops the application).
2. Re-initialize the GIDs by reassigning the IP address. The GIDs will be refreshed to 0, 1, 2, 3 (GID=3 is used again for IPv4 RoCEv2 since no application is holding the QP anymore and no GID index is kept pending for deletion).

How to Avoid the Issue Even When Link Goes Down or Flaps

- Avoid using hard-coded GID number (application dependent). For example, when running RCCL instead of hard-coding the GID with `NCCL_IB_GID_INDEX=3` , use the parameter `NCCL_IB_ROCE_VERSION_NUM=2` (for RoCEv2).
- When implementing an application using libibverbs, do not assume a specific GID for a specific type of traffic. Use `ibv_query_gid()` .
- Automate detection of GID change, e.g. listen for async events (`IBV_EVENT_GID_CHANGE`) - see “How to detect GID change” above.
- Prevent removal or change of the IP address when the link goes down
  - RHEL
    - Once you define the IP address in `/etc/sysconfig/network-scripts/ifcfg-*` , the IP address is kept all the time even when the link goes down (or flaps).
    - Because there is no IP address change, there is no GID table refresh so the GID table is kept intact. The GIDs remain 0, 1, 2, 3.
  - Ubuntu
    - Ubuntu network manager always removes the IP even if it is statically defined in netplan. Because the IP is changed during link down (the IP is removed), it triggers the GID reassignment and this is the cause of GID=4 (assuming prior active QP is still holding a reference to GID=3).
    - One approach is to ignore the link down event so networkd doesn’t take the link down and remove the IP. Sample netplan snippet:

      ```
      network:
      version: 2
      renderer: network
      ethernets:
      eth0:
      ignore-carrier: true
      addresses:
      - 192.168.50.1/24
      ```

    - Other approach is to update network to keep static configuration even upon link down. Sample snippet:

      ```
      /etc/systemd/network/10-static-net.network
      [Network]
      KeepConfiguration=static
      …
      ```

- Schedule maintenance window to restore the initial table (see “How to recover the initial GID index table above).

Note: Link flaps are one of the main reasons for GID index changes. The root cause of the link flaps needs to be addressed in the first place since they could lead to various other errors. Even if the GID table remains intact, the duration of the link flap can disturb ongoing RoCE traffic and cause RoCE transport timeouts and retry attempts.

Unable to get a stable link speed when connecting a BCM957508-P2100G and a BCM957508-N2100G

An incorrect link speed ID is reported when the Ethernet network adapters are connected back-to-back. This occurs when **Media Auto Detect** is enabled. Disable **Media Auto Detect** on both ports on at least one adapter. For example, use the following command to disable it on port 0:

```
niccli -i 1 nvm --setoption media_auto_detect --scope 0 --value 0
```

PPP Over GRE Packet Drops

On BCM5741X and BCM5750X adapters, the hardware parser is enabled by default. A hardware parsing limitation may fail to parse some tunnel packet types and lead to packet drops when using PPP over GRE. To avoid these unintended packet drops, all unnecessary tunnel parsing should be disabled in hardware when being used for PPP over GRE. To address this PPP over GRE issue, an option is available in the NICCLI tool to Enable/Disable GRE.

To retrieve the state of GRE tunnel offload, use the following NICCLI command:

```
niccli -i <index> tunnel --cfg --gre_tunnel_offload --show
```

To set the GRE tunnel offload, use the following NICCLI command:

```
niccli -i <index> tunnel --cfg --gre_tunnel_offload --set --state <enable/disable>
```

Note: The **state** field is used to enable (1) or disable (0) the non-udp port based GRE tunnel offload.

If the user changes the GRE tunnel configuration from NICCLI tool, it is not persistent when the system reboots:

To make this change persistent, there is an option introduced in the **Advanced Driver Property** page **GRE Parsing**. Configure the function-specific GRE tunnel configuration using this parameter which is persistent when the system reboots.

Limitations:

GRE Parsing should be disabled on all the functions of the card to disable **GRE Parsing** at the hardware level. If **GRE Parsing** is enabled on any one function of the card, then global **GRE Parsing** is enabled at the hardware level. Global **GRE Parsing** can be queried using the NICCLI tool.

VLAN Stripping/Acceleration in the Hardware is Enabled by Default in the Driver

It may be required to disable VLAN Stripping/Acceleration when using NVM while running software switch applications (like OVS) or any other software forwarding elements that might offload VLAN push/pop.

What Loopback Modes are Supported on the BCM957608?

The following table shows the loopback modes that are supported by the BCM957608.

Table 140: Supported Loopback Modes

Loopback Mode	Supported	Notes
PHY Local	Yes – for single-port and dual-port 400G cards.	When the TX packet from the host is looped back to the RX, all lanes of the port should be enabled for loopback.
	No – for 4-port 400G cards.	
PHY Remote	Yes – for single-port and dual-port 400G cards.	When the RX packet from the remote partner is looped back to the TX, all lanes of the port should be enabled for loopback.
	No – for 4-port 400G cards.	
MAC Local	No	Not supported for BCM957608.

Loopback Mode	Supported	Notes
External	Yes	Use an external loopback dongle to route the TX from the host back to the RX.

Other Broadcom Resources

Provides links to additional resources for Ethernet adapters available from Broadcom.

Webinar Series on Ethernet Adapters

This section provides links to short videos on topics associated with Ethernet adapters.

Ethernet NIC Technology Protocols, and Standards

Ethernet NIC Performance Tuning and Capabilities

Multi-Host, Multi-Root, and SR-IOV

White Papers

This section provides links to white papers on topics associated with Ethernet adapters.

- Introduction to Thor Congestion Control for RoCE
- NetXtreme® E-Series: Industry's Most Secure Ethernet Adapters

Documentation Legal Notice


BROADCOM®